

Oracle Fusion Middleware 11g: Build Applications with Oracle Forms

Volume I • Student Guide

D61530GC10

Edition 1.0

December 2009

DXXXX

ORACLE®

Author

Pam Gamer

**Technical Contributors
and Reviewers**

Glenn Maslen

Duncan Mills

Grant Ronald

Editors

Raj Kumar

Amitha Narayan

Publishers

Jayanthi Keshavamurthy

Veena Narasimhan

Giri Venugopal

Copyright © 2009, Oracle. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

I Introduction

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Identify the course objectives
- Identify the course content and structure

ORACLE

I - 2

Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

This lesson introduces you to the *Oracle Fusion Middleware 11g: Build Internet Applications with Oracle Forms* course:

- The objectives that the course intends to meet
- The topics that it covers
- How the topics are structured over the duration of the course

Course Objectives

After completing this course, you should be able to do the following:

- Create Forms modules including components for database interaction and GUI controls
- Display Forms modules in multiple windows and a variety of layout styles
- Test Forms modules in a Web browser
- Debug Forms modules in a three-tier environment

ORACLE

I - 3

Copyright © 2009, Oracle. All rights reserved.

Course Objectives

In this course, you learn to build, test, and deploy interactive Internet applications. Working in a graphical user interface (GUI) environment, you learn how to create and customize forms with user input items such as check boxes, list items, and radio groups. You also learn how to modify data access by creating event-related triggers, and you display Forms elements and data in multiple canvases and windows.

Course Objectives

- Implement triggers to:
 - Enhance functionality
 - Communicate with users
 - Supplement validation
 - Control navigation
 - Modify default transaction processing
 - Control user interaction
- Reuse objects and code
- Link one Forms module to another

ORACLE

Course Content

Day 1

- Lesson 1: Introduction to Oracle Forms Builder and Oracle Forms Services
- Lesson 2: Running a Forms Application
- Lesson 3: Working in the Forms Environment
- Lesson 4: Creating a Basic Forms module
- Lesson 5: Creating a Master-Detail Form
- Lesson 6: Working with Data Blocks and Frames

ORACLE

I - 5

Copyright © 2009, Oracle. All rights reserved.

Course Content

The lesson titles show the topics that are covered in this course, and the usual sequence of lessons. However, the daily schedule is an estimate, and may vary for each class.

Course Content

Day 2

- Lesson 7: Working with Text Items
- Lesson 8: Creating LOVs and Editors
- Lesson 9: Creating Additional Input Items
- Lesson 10: Creating Noninput Items

ORACLE

Course Content

Day 3

- Lesson 11: Creating Windows and Content Canvases
- Lesson 12: Working with Other Canvas Types
- Lesson 13: Introduction to Triggers
- Lesson 14: Producing Triggers
- Lesson 15: Debugging Triggers

ORACLE

Course Content

Day 4

- Lesson 16: Adding Functionality to Items
- Lesson 17: Run-Time Messages and Alerts
- Lesson 18: Query Triggers
- Lesson 19: Validation
- Lesson 20: Navigation

ORACLE

Course Content

Day 5

- Lesson 21: Transaction Processing
- Lesson 22: Writing Flexible Code
- Lesson 23: Sharing Objects and Code
- Lesson 24: Using WebUtil to Interact with the Client
- Lesson 25: Introducing Multiple Form Applications

ORACLE

Summary

In this lesson, you should have learned to:

- Identify the course objectives
- Identify the course content and structure

ORACLE

I - 10

Copyright © 2009, Oracle. All rights reserved.

Summary

In this lesson you learned about the objectives of the *Oracle Fusion Middleware 11g: Build Applications with Oracle Forms*. You learned the topics that it covers and how those topics are structured over the duration of the course.

1

Introduction to Oracle Forms Builder and Oracle Forms Services

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to describe the following:

- The components of Oracle Fusion Middleware 11g
- The features and benefits of Oracle Forms Services and Oracle Forms Builder
- The architecture of Oracle Forms Services
- The course application

ORACLE

1 - 2

Copyright © 2009, Oracle. All rights reserved.


Lesson Aim

This course teaches you how to build effective and professional applications by using Oracle Forms Builder.

This lesson identifies the key features of Oracle Fusion Middleware 11g, Oracle Forms Services, Oracle Forms Builder, and the course application model and contents.

There are many terms used in this course that may be unfamiliar to you. For a glossary containing definitions of many of these terms, see <http://www.oracle.com/glossary>.

Web Computing Solutions

Application Type and Audience	Product Approach	Oracle Products
<i>Enterprise applications, business developers</i>	<i>Declarative</i>	<i>Oracle JDeveloper and ADF Oracle Forms</i> 
Java components, component developers	Two-way coding, Java and JavaBeans	Oracle JDeveloper
Self-service applications & content management, Web site developers	Browser-based, dynamic HTML	Oracle WebCenter
Reporting and analytical applications, MIS and business users	Dynamic Web reporting, Drill, analyzing, forecasting	Oracle Reports and Oracle Discoverer

ORACLE

1 - 3

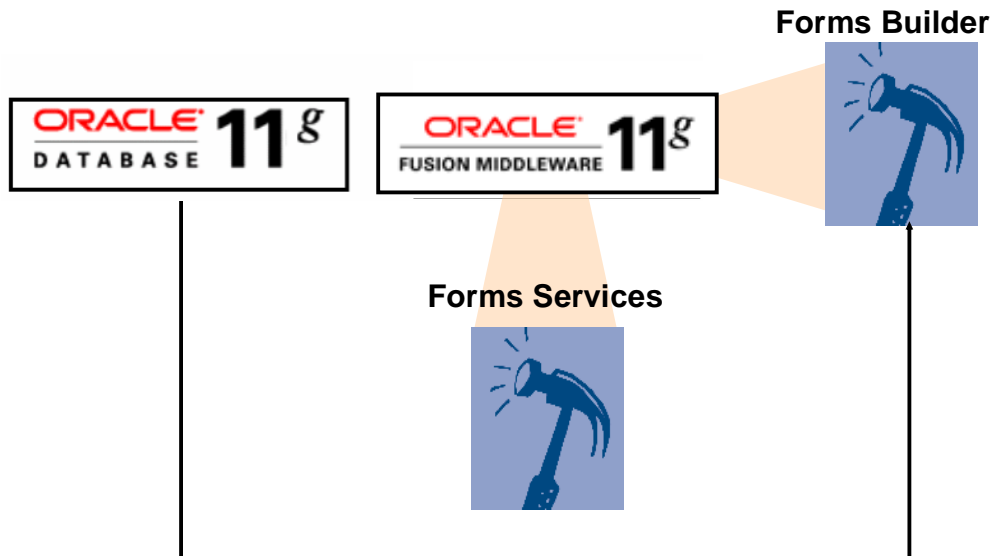
Copyright © 2009, Oracle. All rights reserved.

Web Computing Solutions

Oracle offers a range of tools and deployment options for Web computing, some of which are shown in the slide and described briefly below. Different types of developers and applications require different toolsets.

- Enterprise application developers need a declarative approach. Oracle JDeveloper's Application Development Framework and Oracle Forms both can provide this solution. This course focuses on how you can use Oracle Forms Builder to rapidly build scalable, high-performance applications for the Web and then deploy the applications with Oracle Forms Services.
- Component developers need different tools and methods. For these developers, Java is the language of choice. Oracle's solution is JDeveloper.
- For Web site developers and content publishers who want to build self-service applications for Web sites, Oracle WebCenter supports the creation of all types of portals, Web sites, and composite applications, and is designed to enable business users to evolve these applications as their business requirements change.
- For management information systems (MIS) developers and end users, there is the Oracle Business Intelligence toolset. Oracle Reports Developer, Oracle Reports Services, and Oracle Discoverer provide the whole range for reporting and analysis facilities.

Oracle 11g Products and Forms Development



Oracle 11g Products and Forms Development

The graphic in the slide depicts the three major components used in developing and deploying Forms applications.

Oracle Database: Manages all your information, such as Word documents, Excel spreadsheets, Extensible Markup Language (XML), and images. Oracle Forms Builder is designed for the Oracle database. It delivers the following services for you natively—services you would otherwise have to code by hand:

- Connects to and maintains a connection to the Oracle database
- Queries and handles a large number of records on demand
- Locks database records on demand
- Generates code that automatically supports multi-user locking scenarios
- Manages inserts, updates, and deletes automatically
- Allows programmatic manipulation of sets of records for a developer
- Communicates transactions efficiently to the database in an atomic fashion, meaning that a transaction is either fully committed or not at all
- Automatically handles communication with database Advanced Queuing queues
- Automatically handles logins when database proxy users are used for deployment

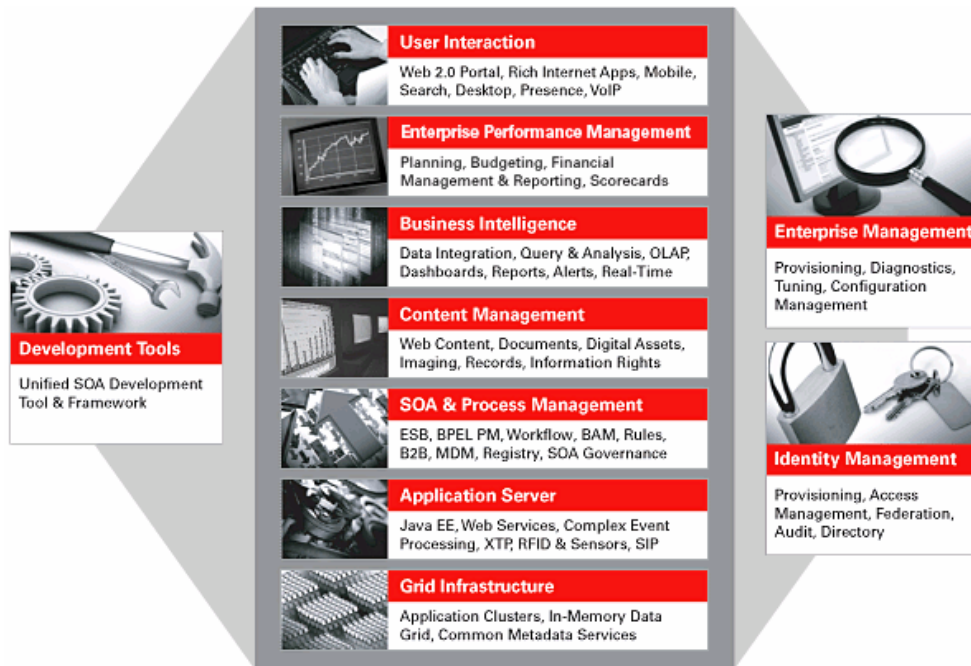
Oracle 11g Products and Forms Development (continued)

The other two components in the slide are part of Oracle Fusion Middleware:

- **Oracle Forms Services:** A comprehensive application framework optimized to deploy Forms applications in a multitier environment. Oracle Forms Services is integrated with Oracle WebLogic Server for running Forms applications.
- **Oracle Forms Builder:** Leverages the infrastructure offered by Oracle Fusion Middleware and Oracle Database, enabling developers to quickly and easily build scalable, secure, and reliable e-business applications. Oracle Forms Builder is included in Oracle Developer Suite, which provides a complete and highly productive development environment for building applications.

Oracle Forms Builder and Oracle Forms Services provide a complete application framework for optimal deployment of Oracle Forms applications on the Web as well as corporate networks. This application framework infrastructure is provided for you, yet you still have the flexibility to leverage the latest technologies within your applications. This allows you to focus on the real value and spend your time thinking about the application business logic and functionality rather than worrying about the application infrastructure.

Oracle Fusion Middleware 11g Architecture



1 - 6

Copyright © 2009, Oracle. All rights reserved.

ORACLE

Oracle Fusion Middleware 11g Architecture

Oracle Fusion Middleware is a standards-based family of products that are often deployed and used in conjunction with one another to develop Java EE applications, providing the benefits of common security, management, deployment architecture, and development tools.

The slide shows an overview of Oracle Fusion Middleware, displaying some of its many features and components, including the following broad categories:

- **Development Tools:** Integrated set of tools for developing SOA applications; includes tools such as JDeveloper and Application Development Framework (ADF), WebCenter Suite, SQL Developer, Forms, Designer, and Reports
- **User Interaction and Enterprise 2.0:** A dynamic, personalized UI layer using Web 2.0 technologies; includes products such as Oracle WebCenter Services and Oracle WebLogic Portal
- **Enterprise Performance Management:** A modular suite of applications supporting strategic and financial management processes; integrates with Oracle Business Intelligence to integrate data from multiple sources and provide dashboards, reporting, and analysis
- **Business Intelligence:** A portfolio of technologies and applications providing an integrated array of query, reporting, analysis, alerting, analytics, desktop integration, and data warehousing tools
- **Content Management:** Robust, scalable solutions for managing all types of content

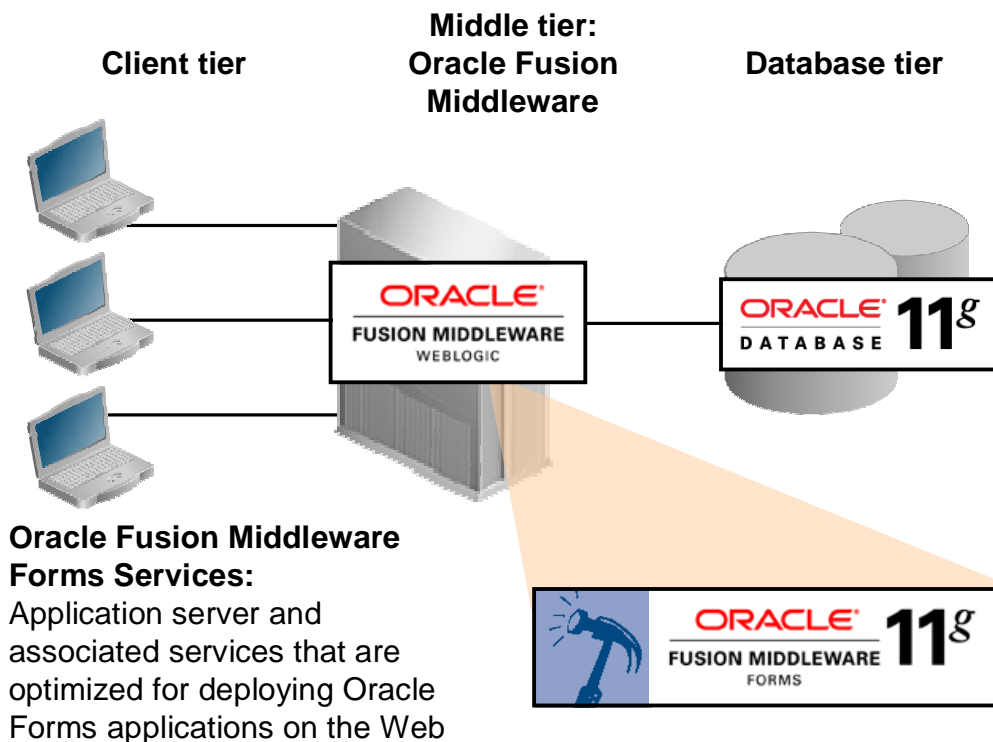
Oracle Fusion Middleware 11g Architecture (continued)

- **SOA and Process Management:** An architecture and tools enabling development of enterprise applications as modular business services that can be easily integrated and reused
- **Application Server:** Oracle WebLogic Server, a complete Java EE 5 and Java SE 6 implementation enabling development and deployment of enterprise applications and services
- **Grid Infrastructure:** An architecture where pools of resources are established, shared, and configured to support varying demands of an enterprise; includes grid control, clustering, and storage management
- **Enterprise Management:** A tool that enables top-down management of applications, middleware, and databases
- **Identity Management:** A set of services providing management of user identities across all enterprise resources, both within and outside the firewall

Oracle Fusion Middleware includes the following major products:

- **Oracle WebLogic Server:** Discussed in the subsequent slides
- **Oracle Metadata Repository:** Contains metadata for system components as well as for configuration of middleware and applications
- **Oracle Identity and Access Management:** An enterprise identity management system
- **Oracle Fusion Middleware Application Components:** Includes Oracle HTTP Server, Oracle Web Cache, Oracle Portal, Oracle Web Services, Oracle Forms Services, and developer tools such as Oracle JDeveloper and Oracle Forms Builder
- **Oracle SOA Suite:** A set of service infrastructure components for creating, managing, and orchestrating services into composite applications and business processes
- **Oracle WebCenter Framework:** An integrated set of components with which you can create social applications, enterprise portals, collaborative communities, and composite applications, built on a standards-based, service-oriented architecture
- **Oracle Business Intelligence:** An integrated solution that addresses all business intelligence requirements

Oracle Forms Services: Overview



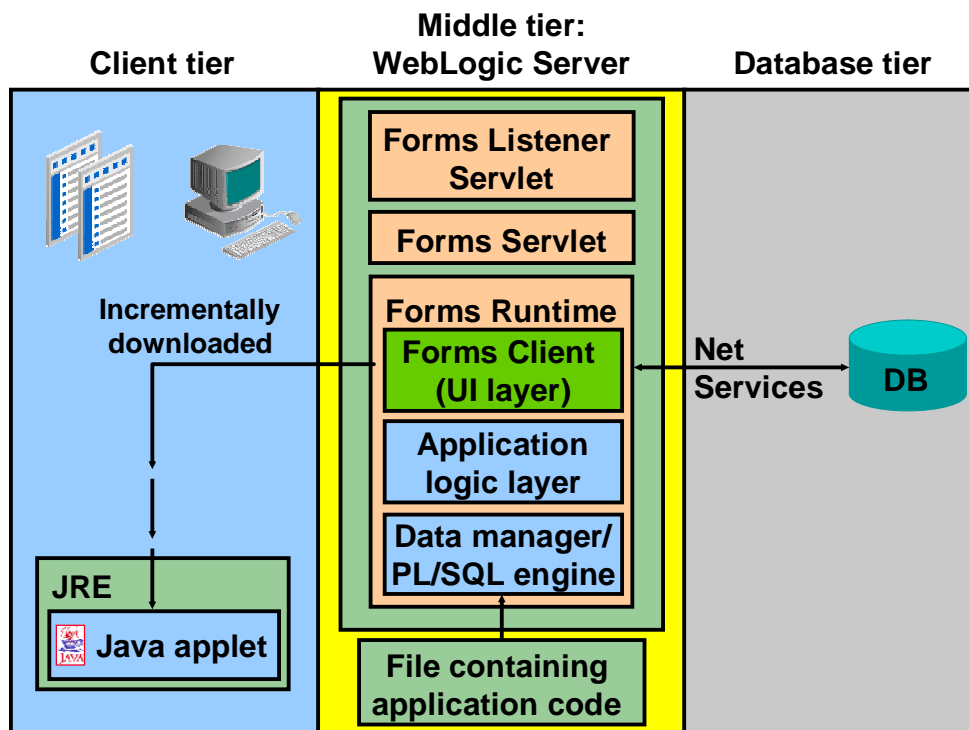
Oracle Forms Services: Overview

Oracle Forms Services is a component of Oracle Fusion Middleware for delivering Oracle Forms applications to the Web. Oracle Forms Services automatically provides the infrastructure that is needed to successfully deliver applications on the Web through built-in services and optimizations.

Oracle Forms Services uses a three-tier architecture to deploy database applications:

- The client tier contains the Web browser, where the application is displayed and used.
- The middle tier is the application server, where the application logic and server software reside.
- The database tier is the database server, where enterprise data is stored.

Forms Services Architecture



ORACLE

1 - 9

Copyright © 2009, Oracle. All rights reserved.

Forms Services Architecture

Forms Services consists of four major components: the Java client (Forms Client), the Forms Listener Servlet, the Forms Servlet, and the Forms Runtime Engine.

When a user runs a Forms session over the Web, a thin, Java-based Forms applet is dynamically downloaded from the application server and automatically cached on the client computer. The same Java applet code can be used for any form, regardless of size and complexity.

Although Forms Services uses a Java applet for displaying the form on the client browser, the developer does not need to know Java in order to develop and deploy a Forms application.

You learn more about the Forms Services components and the process of starting a Forms session in the lesson titled "Running a Forms Application."

Oracle Forms Builder: Overview

Oracle Forms Builder:

- Is a productive development environment for Web-deployed business applications
- Provides for:
 - Data entry
 - Queries



What Is Oracle Forms Builder?

Oracle Forms Builder is a productive development environment for building enterprise-class, scalable database applications for the Web. Oracle Forms Builder provides a set of tools that enable business developers to easily and quickly construct sophisticated database forms and business logic with a minimum effort.

Oracle Forms Builder uses powerful declarative capabilities to rapidly create applications from database definitions that leverage the tight integration with the Oracle database. The toolset leverages Java technology, promotes reuse, and is designed to allow developers to declaratively build rich user interfaces. Developer productivity is further increased through a single integrated development environment that enables distributed debugging across all tiers, utilizing the same PL/SQL language for both server and client.

Oracle Forms 11g: Key Features

- Tools for rapid application development
- Use of PL/SQL scripting language
- Application partitioning
- Extended scalability
- Object reuse
- Integration

ORACLE

1 - 11

Copyright © 2009, Oracle. All rights reserved.

Oracle Forms 11g: Key Features

Tools for rapid application development: You can create and modify applications with little or no code using wizard-based rapid application development and built-in commands.

Use of PL/SQL scripting language: You can write and debug PL/SQL code from within Forms Builder.

Application partitioning: You can place individual PL/SQL program units on the database server or in the application, whichever is most suitable. You can drag objects between modules and the database server.

Extended scalability: The multitiered architecture enables you to scale applications from a single user to tens of thousands of users, with no changes to the application. You can use server functionality, such as array data manipulation language (DML), database cursors, or bind variables, to improve scalability.

Object reuse: Oracle Forms Builder offers an inheritance model that facilitates the inheritance of attributes and code from one object to another and from one application to another, through subclassing and object libraries.

Integration: You can integrate existing Oracle Forms applications to take advantage of Web technologies and Service-Oriented Architecture (SOA).

Oracle Forms 11g: New Features

- For development:
 - Integration with external events
 - JavaScript integration
 - New built-ins to support database proxy users
 - Events for Pluggable Java Components
- For configuration and deployment:
 - Enterprise Manager integration improvements
 - Tracing improvements
 - Support for Oracle Diagnostic Logging
 - Using Java Controller with Reports called from Forms
 - Use of database proxy users

ORACLE

1 - 12

Copyright © 2009, Oracle. All rights reserved.

Oracle Forms 11g: New Features

Oracle Forms 11g includes several new features for both development and deployment:

- **Development features:**
 - Integration with external events by using Oracle database Advanced Queuing
 - Integration with JavaScript, giving the ability to call into Forms from JavaScript and also to invoke JavaScript code from Forms
 - New built-ins for support of using database proxy users
 - Ability to dispatch events from Pluggable Java Components
- **Configuration and deployment features:**
 - New Enterprise Manager user interface and functionality
 - Tracing improvements that enable logging of called PL/SQL functions and procedures with their parameter names, types, and values
 - Support for Oracle Diagnostic Logging (ODL), which is Oracle's standardized logging architecture
 - Ability to use the Java Virtual Machine (JVM) controller when integrating with Oracle Reports
 - Ability to use database proxy users with very few limited privileges for applications with high security requirement or many users

Although these features are not covered in this basic Forms course, you may obtain more information from the sources mentioned in the next slide.

Obtaining More Information

You can obtain further information from the following sources:

- [Oracle Fusion Middleware Administrator's Guide 11g:](http://download.oracle.com/docs/cd/E12839_01/core.1111/e10105/toc.htm)
http://download.oracle.com/docs/cd/E12839_01/core.1111/e10105/toc.htm
- [Oracle Fusion Middleware Forms Services Deployment Guide 11g:](http://download.oracle.com/docs/cd/E12839_01/web.1111/e10240/toc.htm)
http://download.oracle.com/docs/cd/E12839_01/web.1111/e10240/toc.htm
- [Forms page on OTN:](http://www.oracle.com/technology/products/forms)
<http://www.oracle.com/technology/products/forms>
- [Forms forum:](http://forums.oracle.com/forums/forum.jspa?forumID=82)
<http://forums.oracle.com/forums/forum.jspa?forumID=82>
- Forms online help

ORACLE

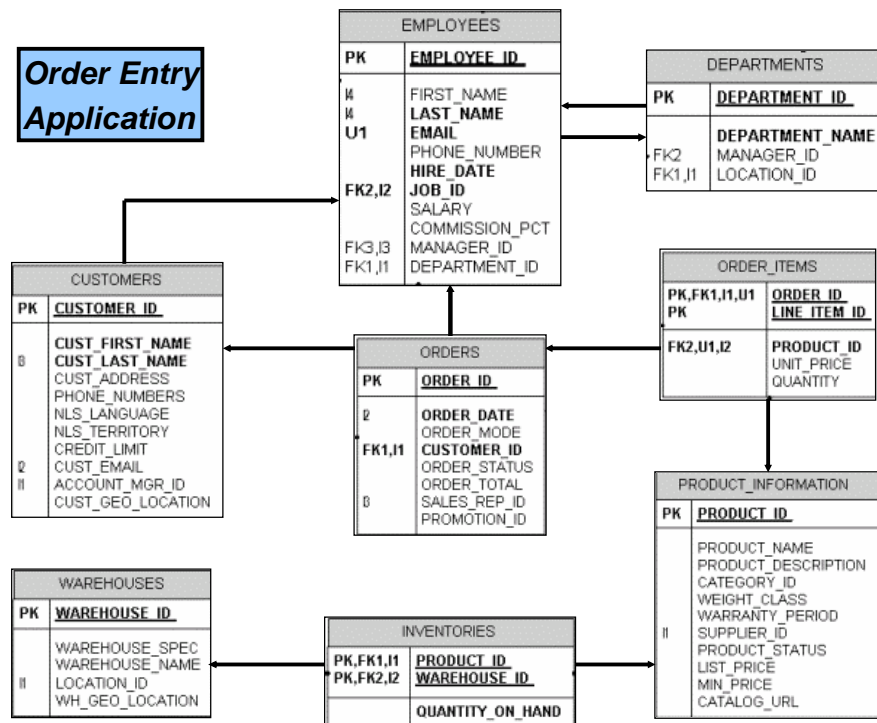
1 - 13

Copyright © 2009, Oracle. All rights reserved.

Obtaining More Information

This course focuses on development. You can learn more about administering Forms Services from the sources mentioned in the slide.

Order Entry Application



1 - 14

Copyright © 2009, Oracle. All rights reserved.

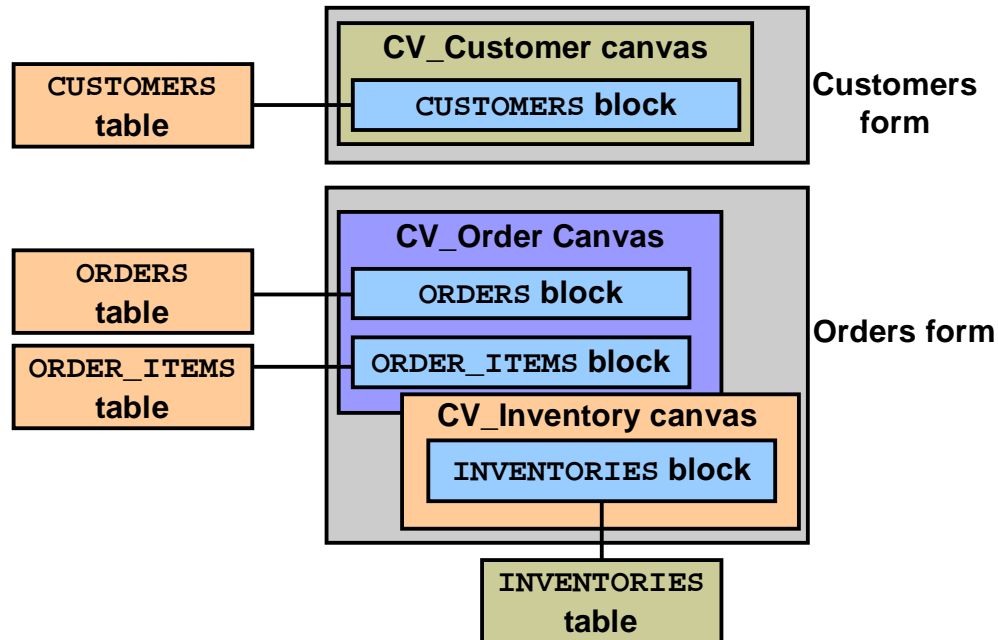
Summit Office Supply Schema

Summit Office Supply is a company that sells office products to customers. Summit has a number of employees in several departments. Some employees are sales representatives who have a relationship with specific customers.

Many products have an associated image, in the form of an image file.

Oracle Fusion Middleware 11g: Build Applications with Oracle Forms 1 - 14

Summit Office Supply Application



ORACLE

1 - 15

Copyright © 2009, Oracle. All rights reserved.

Summit Office Supply Application

The following example of a Forms Builder application will familiarize you with the main run-time facilities of the product. You will also build your own version of this application during the practices in this course.

The Summit company produces a range of office supplies that they sell to businesses and individuals (their customers). The Summit application is an order-entry system that maintains customer details, their orders, and the available stock (inventory).

The application consists of two main forms:

- **Customers form:** The Customers form facilitates queries on existing customers and the insertion, update, or deletion of customer records. When a customer is selected, the user can open the Orders form to enter or view orders for that customer. The form consists of a single block, the **CUSTOMERS block**, which is a single record block whose base table is **CUSTOMERS**.

Summit Office Supply Application (continued)

- **Orders form:** Opened from the Customers form, the Orders form displays orders for a customer and the line items that belong to each order. Orders may also be created, modified, or deleted in this form. You can also display the stock available on the ordered products. The form consists of three blocks:
 - **ORDERS block:** The ORDERS block is a single-record master block for the form. The base table is ORDERS, but the block also displays associated information from other tables, such as the name of the customer.
 - **ORDER_ITEMS block:** The block is the related detail block for an order, showing its line items and the products ordered. This is a multirecord block whose items are on the same canvas as those in the ORDERS block. The base table of the Order_Items block is ORDER_ITEMS, but the block displays information from other tables, such as the product description.
 - **INVENTORIES block:** The INVENTORIES block is a multirecord block showing warehouse stock for a product. Its base table is the INVENTORIES table. Its items are on a separate canvas, which is assigned to its own window. This block is linked to the current product in the ORDER_ITEMS block, but the two blocks can operate independently.

Summary

In this lesson, you should have learned that:

- Oracle Fusion Middleware 11g provides services for building and deploying Web applications
- Oracle Forms Services provides:
 - Optimized Web deployment of Forms applications
 - Rich Java UI without Java coding
 - Generic Java applet to deploy any Forms application
- Oracle Forms Services consists of:
 - Forms Client
 - Forms Servlet
 - Forms Listener Servlet
 - Forms Runtime Engine

ORACLE

1 - 17

Copyright © 2009, Oracle. All rights reserved.

Summary

Oracle Fusion Middleware provides a variety of services for building and deploying Web applications, including Oracle HTTP Server (OHS), Reports Services, and Forms Services.

Oracle Forms Services, a component of Oracle Fusion Middleware 11g, provides for the Web deployment of Forms applications with a rich Java user interface. It uses the same generic applet for any form.

The components of Oracle Forms Services all play a role in running an application. These components are the Forms Client (Java applet), the Forms Servlet, the Forms Listener Servlet, and the Forms Runtime Engine.

Summary

- Benefits of Oracle Forms Builder include rapid application development, application partitioning, flexible source control, extended scalability, and object reuse
- The course application is a customer and order entry application for Summit Office Supply

ORACLE

1 - 18

Copyright © 2009, Oracle. All rights reserved.

Summary (continued)

Oracle Forms Builder enables you to develop Forms applications. Benefits of Oracle Forms Builder include:

- **Rapid application development:** Creating and modifying applications with little or no code
- **Application partitioning:** Dragging objects between modules and the database server
- **Flexible source control:** Integration with Software Configuration Manager (SCM)
- **Extended scalability:** Use of server functionality such as array DML, database cursors, or bind variables
- **Object reuse:** Subclassing, object libraries

2

Running a Forms Application

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Start WebLogic Server
- Describe the run-time environment
- Describe the elements in a running form
- Navigate a Forms application
- Describe the two main modes of operation
- Run a form in a Web browser:
 - Retrieve both restricted and unrestricted data
 - Insert, update, and delete records
 - Display database errors

ORACLE

2 - 2

Copyright © 2009, Oracle. All rights reserved.

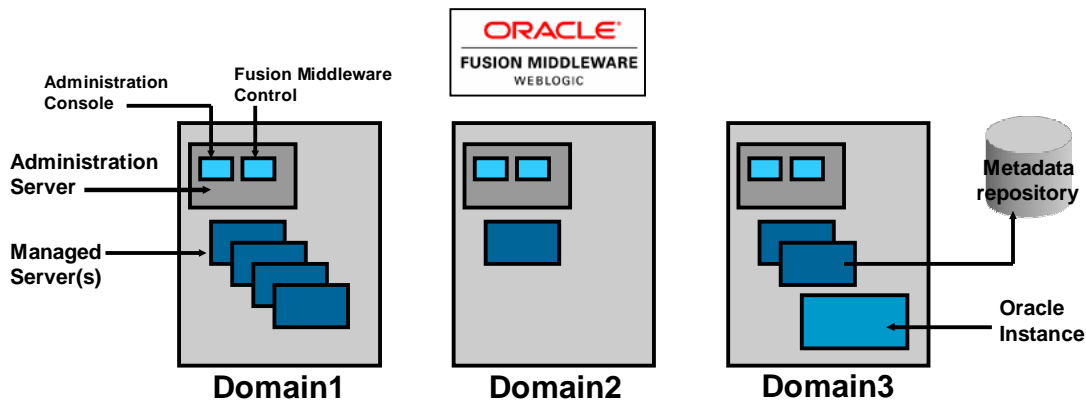
Lesson Aim

It is helpful to understand the environment of the form operator before designing and building your own applications. In this lesson, you run an existing application on the Web in order to familiarize yourself with the run-time interface of Oracle Forms Builder.

What Is Oracle WebLogic Server?

Oracle WebLogic Server:

- Is Oracle's application server for running and administering Forms (and other) applications
- Defines manageable environments called domains
- Enables testing of Forms applications within Forms Builder



What Is Oracle WebLogic Server?

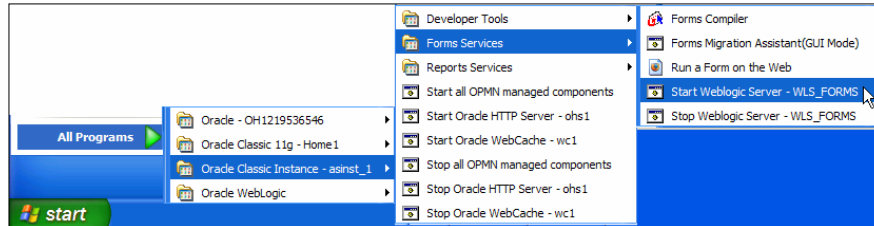
Oracle WebLogic Server is a scalable, enterprise-ready Java Platform, Enterprise Edition (Java EE) application server. The WebLogic Server infrastructure supports the deployment of many types of distributed applications and is an ideal foundation for building applications based on Service-Oriented Architecture (SOA). SOA is a design methodology aimed at maximizing the reuse of application services.

The graphics in the slide show that the WebLogic Server can define multiple environments called domains. Each domain consists of one Administration Server and one or more Managed Servers. Depending on the components that are installed, you may also have an Oracle Instance, and possibly a metadata repository if the installed components require one.

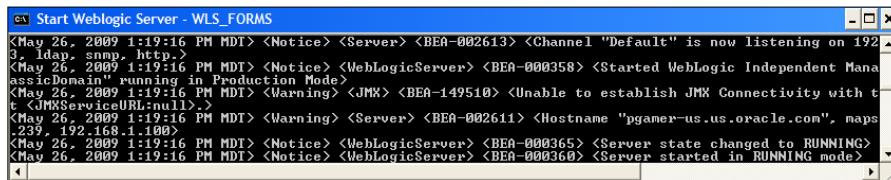
WebLogic Server provides a servlet container that is ideally suited to run Forms applications. The servlet container calls the servlet's methods and provides services that the servlet needs when running. When a request for a Forms application is routed to WebLogic Server, WebLogic Server performs the following actions:

1. If an instance of the servlet does not exist, it loads the servlet class, then instantiates and initializes an instance of the servlet class.
2. It invokes the servlet, passing request and response objects. The request object contains information about the client, request parameters, and HTTP headers. The response object returns the servlet's output to the client.

Starting and Stopping Oracle WebLogic Managed Servers



- Run `startManagedWebLogic.cmd` or `stopManagedWebLogic.cmd` (can run from the Start menu).
- You can minimize, but do not close, the command window.



ORACLE

Starting and Stopping Oracle WebLogic Managed Servers

A managed server is a WebLogic Server instance that runs deployed applications. When you install and configure Oracle Forms, a managed server is created named `WLS_FORMS`. In order to run a form, you first must start this managed server. You can start a managed server even if the Administration Server (needed for configuring Forms Services) is not running.

To start or stop the `WLS_FORMS` managed server on Windows, execute the appropriate batch file, either `startManagedWebLogic.cmd` or `stopManagedWebLogic.cmd`, located in the `user_projects\domains\ClassicDomain\bin` subdirectory of the Fusion Middleware home directory. You must pass `SERVER_NAME` and `ADMIN_URL` command-line arguments to these scripts; for example:

```
startManagedWebLogic.cmd WLS_FORMS t3://myhost.com:7001
```

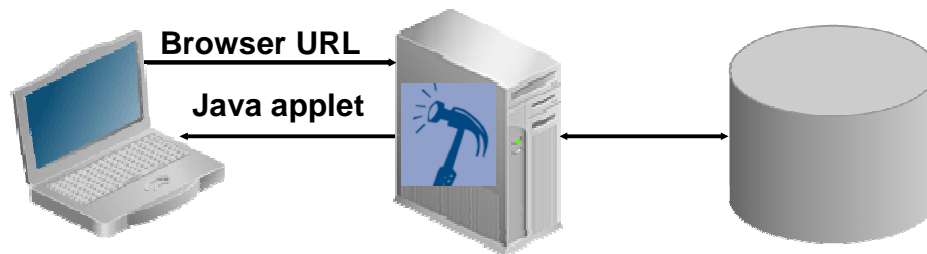
Note: The batch files on Linux systems end with the `.sh` extension rather than `.cmd`.

Alternatively, you can call these batch files from the Windows Start menu as shown in the top screenshot in the slide: All Programs > Oracle Classic Instance > Forms Services > Start [Stop] WebLogic Server – `WLS_FORMS`. These menu items already have the command-line arguments. You will need to input the username and password for starting the Oracle WebLogic Server, which were specified at the time of installation.

You can minimize, but do not close, the command window, as shown in the bottom screenshot.

Running a Form

Oracle Forms Services deployment:



ORACLE

2 - 5

Copyright © 2009, Oracle. All rights reserved.

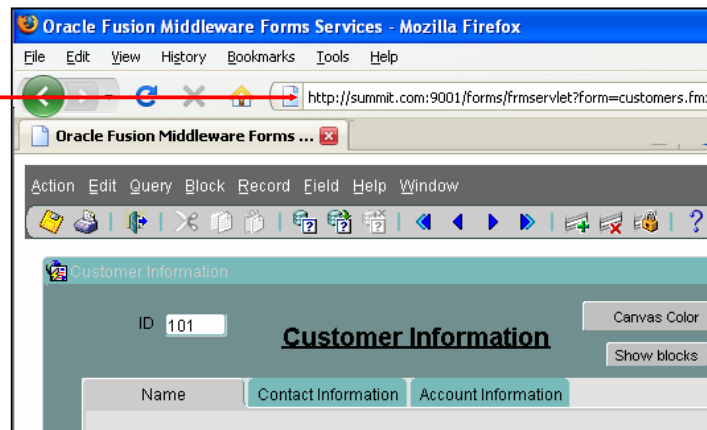
Running a Form

Deploying Forms applications to the Web is implemented by the three-tier architecture of Oracle Fusion Middleware. Application logic and the Forms Runtime Engine reside on the middle tier application server. All trigger processing occurs on database and application servers, whereas user interface processing occurs on the Forms Client. End users can run Forms applications in a Web browser.

Users request an application in their Web browsers by entering a URL that points to the application. Forms Services then generates an HTML file that downloads a Java applet to the client machine. This small applet is capable of displaying the user interface of any form, while the application logic is executed on the middle tier.

The graphics in the slide depict a client computer at the left and a middle tier computer in the middle, with Forms Services running. The middle tier computer communicates with the database shown at the right of the slide. The client sends an HTTP request in the form of a URL to the middle tier, which downloads a Java applet to the client.

Running a Form: Browser



```
http://summit.com:9001/forms/frmservlet  
?form=<formname>.fmx  
&userid=<username>/<password>@<database>  
&buffer_records=NO&debug_messages=NO&array=YES  
&query_only=NO
```

ORACLE

2 - 6

Copyright © 2009, Oracle. All rights reserved.

Running a Form: Browser

The graphic in the slide shows a user accessing the application. The URL to invoke an application must have the following format:

`http://<host>.<domain>:<port>/forms/frmservlet[?<parameter list-value pairs separated by "&">]`. Optional portions of the URL are enclosed in brackets.

Summit's URL consists of the following components:

Protocol	http
Host and domain	summit.com
Port for WebLogic Server or Oracle HTTP Server	9001 default for WebLogic Server (used for testing from Forms Builder) 8888 default for Oracle HTTP Server (more commonly used for production)
Forms Servlet Alias or static HTML file	/forms/frmservlet
Parameters: This section begins with "?"; parameters separated by "&" (can be specified in the URL or taken from configuration file).	form=customers.fmx userid=<username>/<password>@<database> buffer_records=NO debug_messages=NO

Java Runtime Environment



- The Forms applet runs in a Java Runtime Environment (JRE) on the client machine.
- Forms uses the JRE provided by the Sun Java Plug-in (JPI).

ORACLE

2 - 7

Copyright © 2009, Oracle. All rights reserved.

Java Runtime Environment

The Web browser can run a Java applet because it provides a JRE. However, the advanced features of the Forms Client require the Sun JavaSoft 1.6.x plug-in (other JREs may be supported in later patch releases) with one of the following browsers:

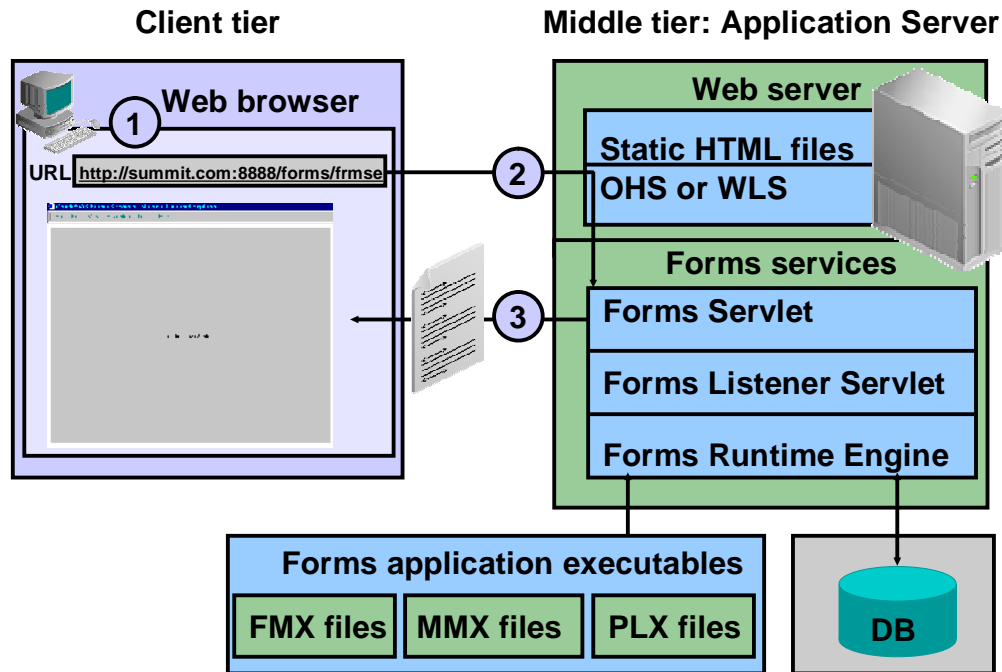
- **IE 7.x:** When a user attempts to run a Forms application with the Internet Explorer browser on a computer that does not have Java 1.6 installed, the user can download the plug-in, and it is automatically installed.
- **Firefox 3.x browser:** When a user attempts to run a Forms application with the Mozilla Firefox browser on a computer that does not have Java 1.6 installed, Firefox does not download the JRE 1.6 plug-in automatically. Instead, Firefox displays the following message: “Additional plugins are required to display this page...”. The workaround is to download the JRE 1.6 plug-in by clicking the Install Missing Plugin link and install it manually, then restart Firefox.

The slide shows a symbol of the Sun JavaSoft Plug-in.

For more information about using JPI, see the following white paper:

Oracle 10gR2 Forms Services – Using Sun’s Java Plug-in

Starting a Run-Time Session

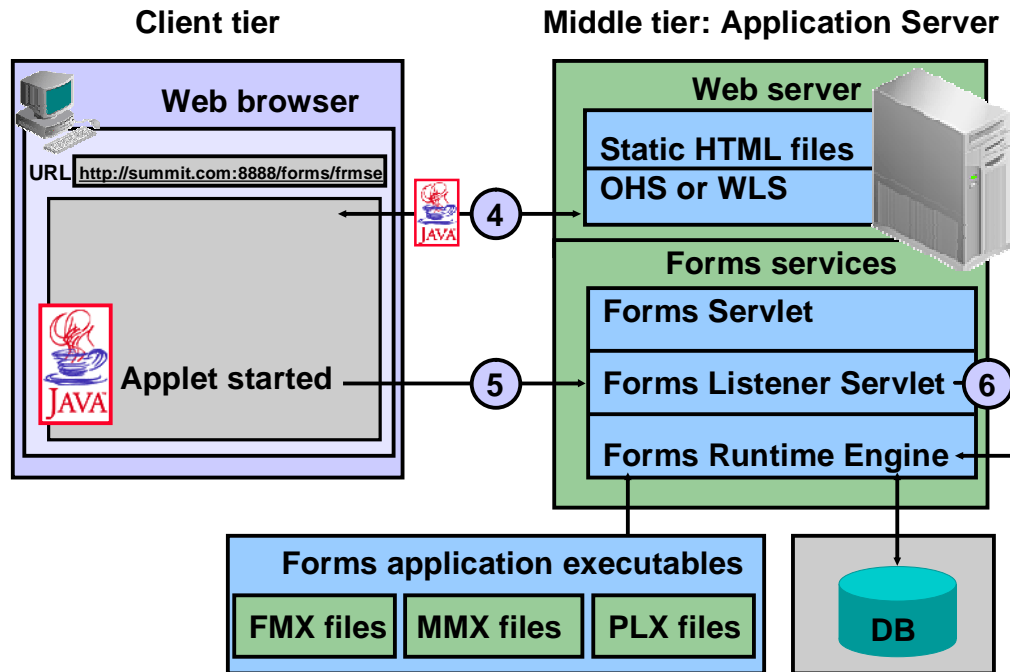


Starting a Run-Time Session

Starting a run-time session involves the following steps, which are depicted as numbered in this and the following slides:

1. The user accesses the URL that indicates that a Forms application should be run.
2. The Oracle HTTP Server (OHS) or the Oracle WebLogic Server (WLS) receives an HTTP request from the browser client and contacts the Forms Servlet.
3. The Forms Servlet dynamically creates an HTML page containing all the information to start the Forms session.

Starting a Run-Time Session



ORACLE

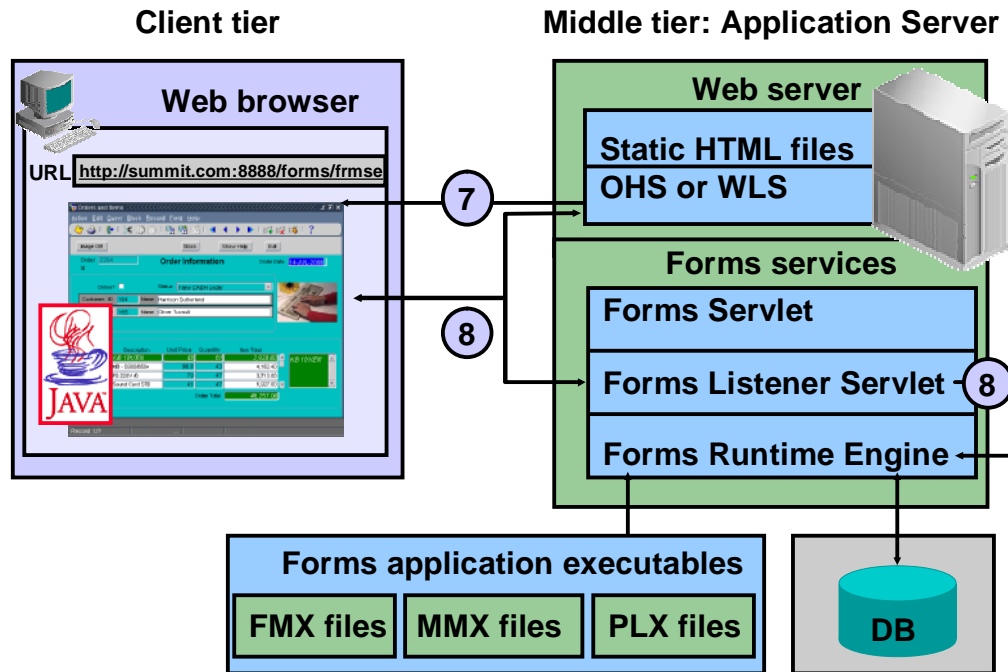
2 - 9

Copyright © 2009, Oracle. All rights reserved.

Starting a Run-Time Session (continued)

4. The Oracle HTTP Server or the Oracle WebLogic Server downloads a generic applet to the client after checking that it has not already been downloaded. The client caches the applet so that it can run future Forms applications without downloading it again.
5. The client applet contacts the Forms Listener Servlet to start the session. The Forms Listener Servlet starts an instance of the Forms Runtime Engine on the Forms Server (middle tier). If included in the HTML file, Forms Runtime command-line parameters (such as form name, user ID and password, database SID, and so on) and any user-defined Forms Builder parameters are passed to the process by the Forms Listener Servlet.
6. The Forms Listener Servlet establishes a connection with the Forms Runtime Engine, which connects to the database if needed and loads application executable files.

Starting a Run-Time Session



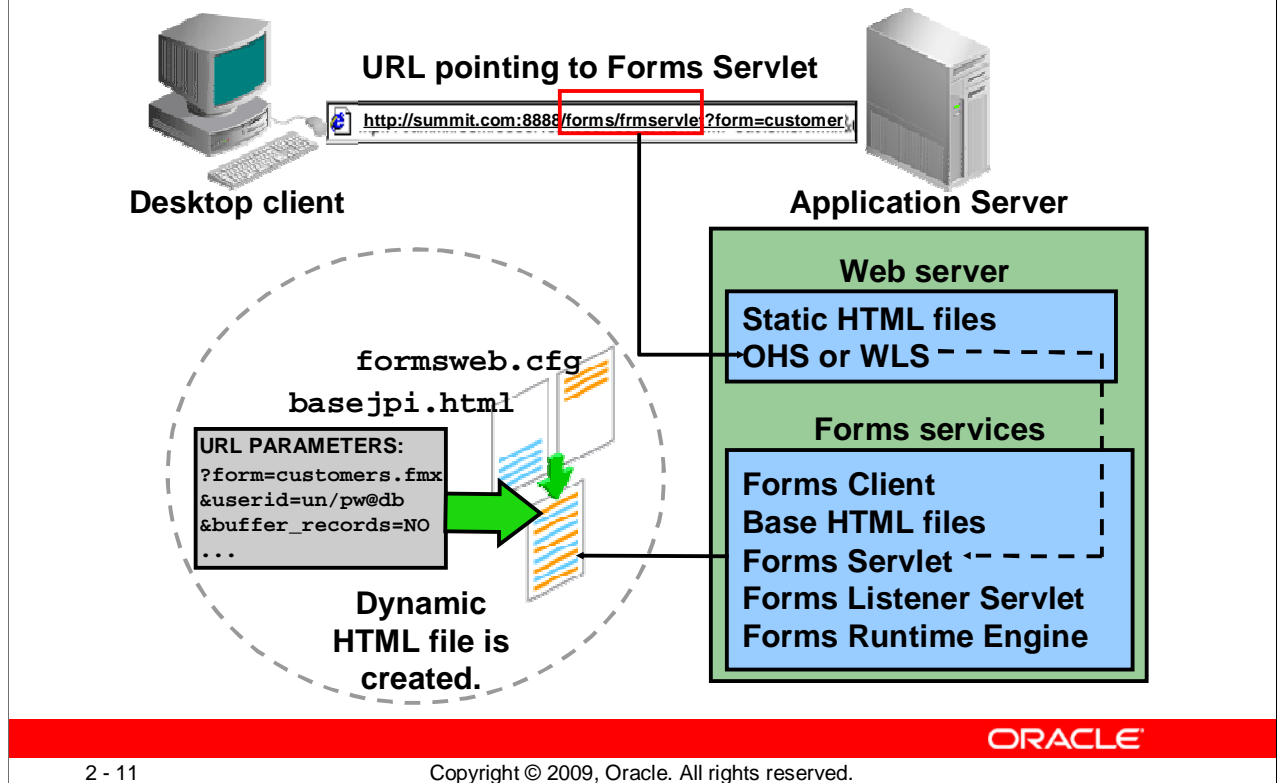
Starting a Run-Time Session (continued)

7. The Forms applet displays the user interface of the application in the main window of the user's Web browser.
8. The Forms Listener Servlet, working through HTTP Server or WebLogic Server, manages communication between the Forms applet and the Forms Runtime Engine.

Technical Note

For more information about Oracle Forms Listener Servlet and the connection process, refer to the *Oracle Forms Services & Oracle Forms Developer 11g Technical Overview* white paper, available on OTN.

Forms Servlet



Forms Servlet

The Forms Servlet is a Java servlet that creates a dynamic HTML file by merging information from the following sources:

- The Forms Web configuration file
- The Forms base HTML file
- The application's URL parameters

The graphics in the slide show:

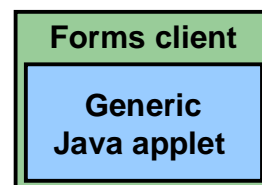
- A desktop client sending a URL to the application server; the URL contains "forms/frmservlet" to invoke the Forms Servlet
- The Web server receiving the request and passing it to the Forms Servlet component of Forms Services
- The Forms Servlet constructing the HTML file by combining parameters from the `formsweb.cfg` file, the `basejpi.html` file, and the parameters that are passed on the URL

Forms Client

- Generic Java applet
- Responsibilities:
 - Displays the form's user interface
 - Processes user interaction back to Forms Services
 - Processes incoming messages from Forms Services



Desktop client



ORACLE

2 - 12

Copyright © 2009, Oracle. All rights reserved.

Forms Client

The Forms Client is a generic Java applet. Forms Services dynamically downloads this applet and automatically caches it on the client machine. The Forms Client consists of a set of Java classes. At startup, only those Java classes that are necessary to initialize the application are downloaded. Additional class files are downloaded dynamically, as needed, to support additional user interface activity.

You do not have to deploy a separate Java applet for each application. The same generic applet is used to run any Forms Services application, regardless of its size and complexity.

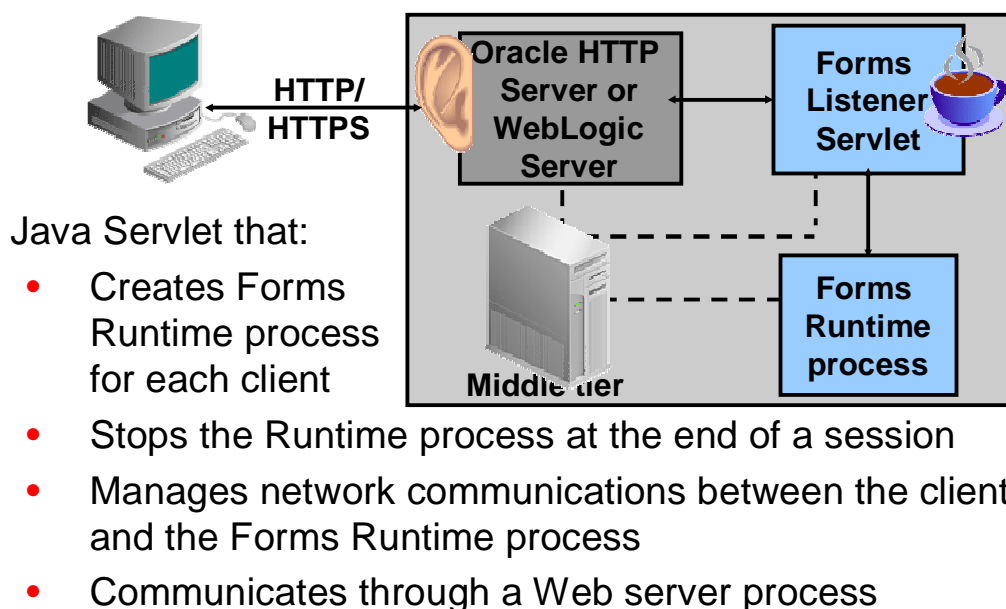
Responsibilities of the Forms Client

The Forms Client represents the user interface layer and has three primary functions:

- To render the Forms Services application display for the user
- To efficiently process user interaction back to Forms Services
- To process incoming messages from Forms Services and translate them into interface objects for the end user efficiently

The slide graphics depict the Forms Client that is downloaded to the desktop client computer in the form of a generic Java applet.

Forms Listener Servlet



Forms Listener Servlet

The Forms Listener Servlet is a Java servlet that runs in a Web server equipped with a servlet engine, such as WebLogic Server. The Web server directs HTTP requests for the Forms Listener Servlet directly to the servlet instances.

The Forms Listener Servlet is in charge of:

- Managing the creation of the Forms Runtime process for each client
- Managing the network communications that occur between the client and its associated Forms Runtime process, through the Web server

This means that the client sends HTTP requests and receives HTTP responses from the Web server process itself. Because the Web server acts as the network endpoint for the client, there is no need to expose additional server machines and ports at the firewall.

The graphics in the slide show that the Forms Listener Servlet creates the Forms Runtime process and manages communication between that process and the desktop client.

Forms Runtime Engine

The Forms Runtime Engine:

- Is a process (`frmweb`) that runs on the Application Server
- Manages application logic and processing
- Communicates with the client browser and the database

ORACLE

2 - 14

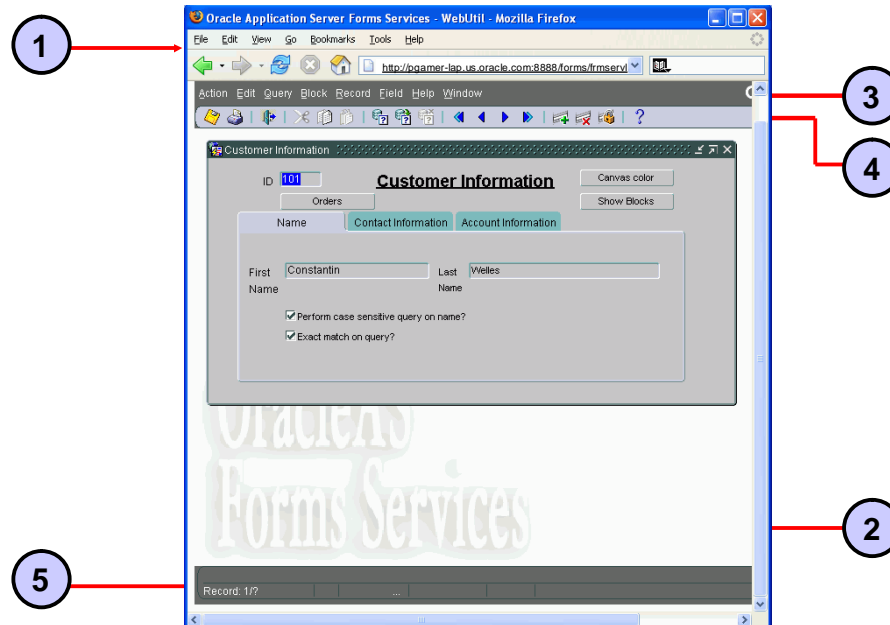
Copyright © 2009, Oracle. All rights reserved.

Forms Runtime Engine

The Forms Runtime Engine is a process on the Application Server that is started by the Forms Listener Servlet. You cannot start the runtime engine directly.

The Forms Runtime Engine handles all the application logic and Forms functionality and executes the code written into the application. It manages requests from the Forms Client and sends metadata to the client to describe the user interface. It connects to and communicates with the Oracle database via Oracle Net Services, the replacement for Net8 and SQL*Net.

What You See at Run Time



What You See at Run Time

At run time, you will see the following components that are pictured and numbered as follows in the screenshot in the slide:

1. Browser window
2. Java applet (contained within the browser window)
3. Default menu (contained within the applet)
4. Menu toolbar (contained within the applet)
5. Console (contained within the applet)

What Is the Default Menu?

The *default menu*, which is part of all Oracle Forms applications, is an alternative to keystroke operations. You can replace or customize the Default menu to introduce your own functionality into a form module.

What Is the Menu Toolbar?

The *menu toolbar* contains buttons corresponding to menu items. At run time, it appears above any user-defined toolbars. It executes the same code as menu items, and it is a shortcut to menu commands that does not duplicate code or effort.

Identifying the Data Elements

The screenshot shows an Oracle Forms application window titled "Orders and Items". It contains several data elements identified by numbered callouts (1-10):

- 1: "Show Blocks" button
- 2: "Order Id" text prompt
- 3: "Online?" checkbox
- 4: "Order Information" boilerplate graphic
- 5: "Status" dropdown menu (set to "New CASH order")
- 6: "Customer: ID 104" display item
- 7: "Name: Harrison Sutherland" display item
- 8: "Exit" button
- 9: "KB 101/EN" image item
- 10: "Credit Limit" radio group (with options Low, Medium, High)

A "Customer Information" dialog box is also visible, showing fields for ID, Name, Contact Information, Account, and Manager ID, along with a "Credit Limit" radio group.

Item#	Id	Description	Unit Price	Quantity	Item Total
1	3106	KB 101/EN	48.00	61	2,9
2	3114	MB - S900/6500	96.80	43	4,1
3	3123	PS 220V /D	79.00	47	3,7
4	3129	Sound Card STD	41.00	47	1,9
Order Total					46,5

Identifying the Data Elements

A Forms application may contain different kinds of data elements that are pictured and numbered in the screenshot in the slide:

1. Prompts
2. Text items
3. Check boxes
4. Boilerplate graphics
5. Boilerplate text
6. Display items
7. List items
8. Push buttons
9. Image items
10. Radio groups

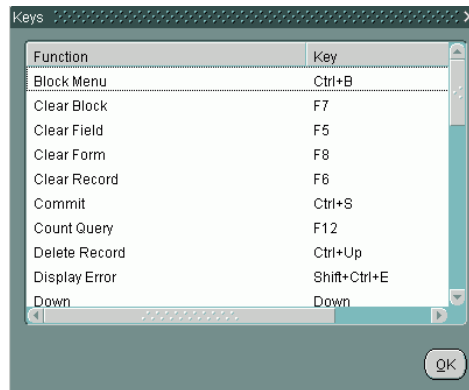
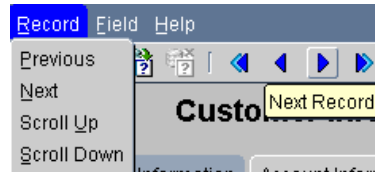
Not pictured:

- Hierarchical tree items
- Chart items
- Custom items

Navigating a Forms Application

Methods of navigation:

- Default menu
- Menu toolbar
- Mouse
- Buttons
- Function keys



ORACLE

2 - 17

Copyright © 2009, Oracle. All rights reserved.

Navigating a Forms Application

Default Menu

The default menu, shown in the top screenshot in the slide, is automatically available in a form, unless it is disabled or replaced with a customized menu. Select options from the menu by using the mouse or function keys. At run time, use the menu to perform the following tasks:

- Move the cursor and navigate between data blocks, records, and items.
- Save or clear all changes.
- Execute queries.
- Insert new records or delete existing records.
- Invoke Help.

Menu Toolbar

You can use the default menu toolbar buttons, also shown in the top screenshot, to perform the following operations, which are also available through the default menu:

- Save all the changes.
- Exit the form.
- Execute queries.
- Navigate between data blocks or records.

Navigating a Forms Application (continued)

- Insert new records or delete existing records.
- Invoke Help to see the properties of an item.

Mouse

You can use the mouse to navigate and to perform many user operations in a bitmapped environment without needing to learn the function keys. Use the mouse to perform the following actions:

- Move the cursor.
- Select from a menu.
- Select from a list of values (LOV).
- Select or deselect a check box.
- Select a button, including an option button.
- Switch to an open window.
- Respond to an alert.
- Scroll records or lines by using a data block or item scroll bar.
- Manipulate a custom item.

Buttons

Web applications often use buttons as a means of navigation. You can click buttons with the mouse. Use buttons to perform the following tasks:

- Move input focus.
- Display a list of values.
- Invoke an editor.
- Invoke another window.
- Commit data.
- Issue a query.
- Perform calculations.
- Exit the form.

Function Keys

In addition to navigating with the mouse, you can move from item to item in sequence with function keys. Use function keys to perform the following tasks:

- Navigate between data blocks, records, and items.
- Execute queries.
- Insert new records or delete existing ones.
- Invoke Help.

To view a list of keys and the functions they perform, select Help > Keys, or press Ctrl + K. The Keys window, shown in the bottom screenshot in the slide, displays the list of keys and functions.

Modes of Operation: Enter-Query Mode

Allows:

- Unrestricted and restricted queries
- Query/Where dialog box
- Record count by using Query > Count Hits

Does not allow:

- Navigation out of the current data block
- Exiting the run-time session
- Certain functions
- Insert, update, delete

ORACLE

2 - 19

Copyright © 2009, Oracle. All rights reserved.

Modes of Operation: Enter-Query Mode

Forms Builder has two main modes of operation: Enter-Query mode and Normal mode. The third mode of operation, Query mode, occurs while Forms is processing a query; the user cannot interact with the form while this query processing is taking place.

Enter-Query Mode

Use Enter-Query mode to enter search criteria for a database query. In Enter-Query mode, your keystrokes are interpreted as search criteria for retrieving restricted data.

You can do the following in Enter-Query mode:

- Retrieve all records.
- Retrieve records by using selection criteria.
- Retrieve records by using the Query/Where dialog box.
- Obtain the number of records that will be retrieved before fetching them from the database by using Query > Count Hits.

You cannot do the following in Enter-Query mode:

- Navigate out of the current block.
- Exit from the run-time session.
- Use certain functions, such as Next Record.
- Insert new records.
- Update existing records.
- Delete records.

Modes of Operation: Normal Mode

Allows:

- Unrestricted queries
- Insert, update, delete
- Commit (Save)
- Navigation out of the current data block
- Exiting the run-time session

Does not allow:

- Restricted queries
- Query/Where dialog box

ORACLE

2 - 20

Copyright © 2009, Oracle. All rights reserved.

Modes of Operation: Normal Mode

Use Normal mode to insert and alter records in the database. In Normal mode, your keystrokes are interpreted as either the entering of new records or the altering of existing ones.

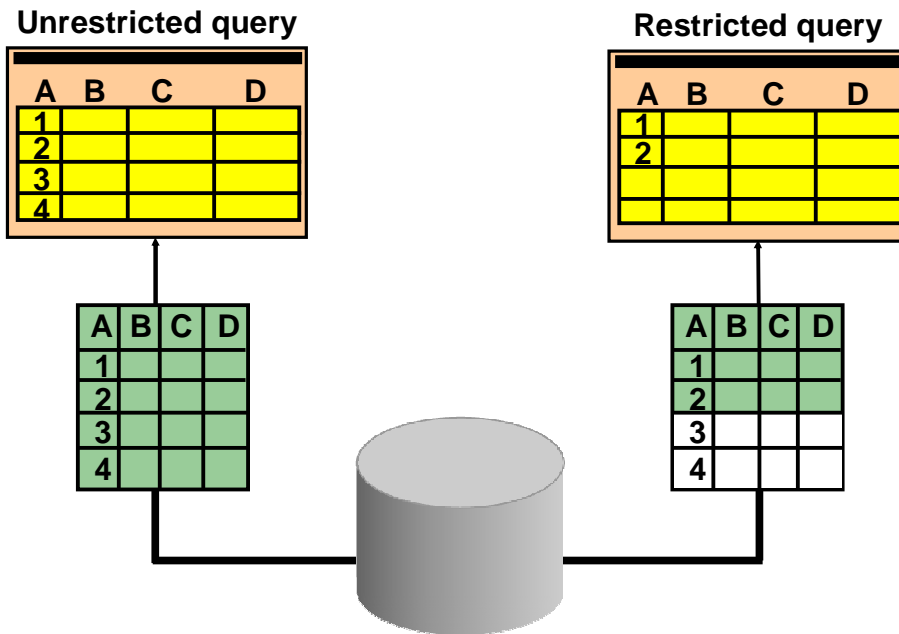
You can do the following in Normal mode:

- Retrieve all records.
- Insert new records.
- Update records.
- Delete records.
- Commit (Save) records.
- Roll back (Clear) records.
- Navigate out of the current data block.
- Exit the run-time session.

You cannot do the following in Normal mode:

- Retrieve a restricted set of records.
- Invoke the Query/Where dialog box.

Retrieving Data



ORACLE

2 - 21

Copyright © 2009, Oracle. All rights reserved.

Retrieving Data

You can use a form module to retrieve information from the database without knowing any SQL syntax. However, if you are an experienced SQL user, you may want to supplement Forms default processing with your own SQL predicates. There are two general types of queries:

Query Type	Description
Unrestricted (global query)	The equivalent of selecting all the rows for all the represented columns from the base table for the queried data block; depicted by the graphics at the left of the slide
Restricted	The equivalent of selecting a restricted set of rows for all the represented columns from the base table for the queried data block; depicted by the graphics at the right of the slide

Performing an Unrestricted Query

You can retrieve unrestricted data by performing one of the following actions:

- Select Query > Execute.
- Press the appropriate function key.
- Click the Execute Query button.

Note: You cannot perform a query while you have unsaved updates, inserts, or deletes. Either save or undo the changes before you continue with the query.

Performing a Restricted Query

- Do not use quotation marks with character and date items.
- The `LIKE` operator is implied with `%` or `_`.
- Use hash (`#`) in front of SQL operators.
- Use Query/Where for complex query conditions.
- Use default date format (DD-MON-RR) in Query/Where.
- Use double quotation marks around literals in Query/Where.

ORACLE

2 - 22

Copyright © 2009, Oracle. All rights reserved.

Performing a Restricted Query

You can use any one of the following methods to perform a restricted query:

- Matching values
- Matching patterns (wildcards)
- A Query/Where dialog box for user entry of SQL predicates

Valid Search Criteria

Item	Criterion	Uses
Order_Id	2356	Exact match
Customer_Id	%04	Implied <code>LIKE</code> operator
Order_Id	#BETWEEN 2350 AND 2360	<code>BETWEEN</code> operator
Order_Status	“CREDIT order paid” selected from pop-up list	Exact match on item value (10)
Sales_Rep_Id	:S	Query/Where dialog box

Performing a Restricted Query (continued)

How to Perform a Restricted Query

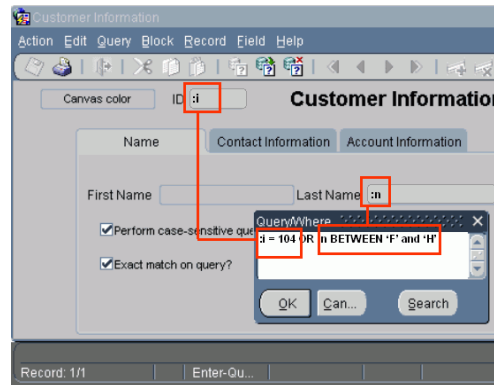
You can perform a restricted query by using the following steps:

1. Perform one of the following:
 - Select Query > Enter.
 - Click the Enter Query button.
 - Press the appropriate function key.
2. Enter-Query displays on the status line.
3. Enter search criteria into appropriate items.
4. Perform one of the following:
 - Select Query > Execute.
 - Click the Execute Query button.
 - Press the appropriate function key.

Note: Forms Builder constructs a select statement by using the AND operator for all the specified conditions.

Using the Query/Where Dialog Box

- Invoke by:
 - Entering
 - `:<variable_name>`
 - `&<variable_name>`
 - `:`
 - `&`
 - Executing query
- Used to write:
 - Complex search conditions
 - Queries with OR predicates
 - The ORDER BY clause



ORACLE

Using the Query/Where Dialog Box

In the Query/Where dialog box, you can enter complex search criteria by using raw SQL. To use the Query/Where dialog box effectively, you should have knowledge of SQL. Use Query/Where to perform the following tasks:

- Write complex search conditions.
- Write queries with OR predicates.
- Order the result of a query.

Forms Builder logically uses the AND operator to append the Query/Where conditions to any other search criteria (including those imposed by the form designer) and constructs a SELECT statement. The Query/Where dialog box does not work in an item whose name differs from the name of its corresponding database column.

To improve security, Forms 10.1.2.0.2 introduced a new run-time environment variable, `FORMS_RESTRICT_ENTER_QUERY`, that is initially set to `TRUE`, which makes it impossible to invoke the Query/Where dialog box. You must set this environment variable to `FALSE` (or comment out the line that sets it to `TRUE`) to enable the use of the Query/Where dialog box. The lesson titled “Working in the Forms Environment” discusses environment variables and how to set them.

Using the Query/Where Dialog Box (continued)

To use the Query/Where dialog box to enter query criteria:

1. Perform one of the following:
 - Select Query > Enter.
 - Click Enter Query.
 - Press the appropriate function key.
2. Enter a colon (:) or an ampersand (&) followed by a unique character variable name in one or more items.
3. Perform one of the following: Select Query > Execute, click Execute Query, or press the appropriate function key.

Note: You can select Query > Count Hits if you simply want to know how many records will match your criteria.
4. The Query/Where dialog box is displayed.
5. Enter the search criteria by using variables, SQL, and logical operators. Click OK.

Note: To perform a query without any variables, enter only the colon (:) or ampersand (&) and execute the query. Doing so also displays the Query/Where dialog box.

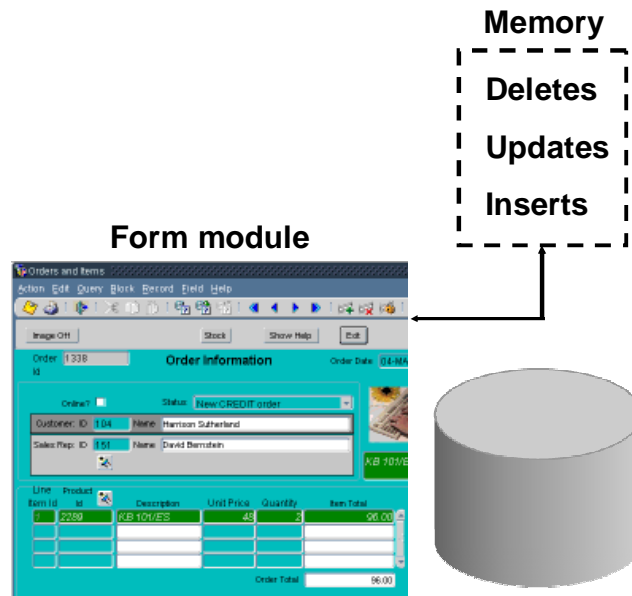
If you enter an ORDER BY at run time, it overrides any ordering defined by the designer.

The screenshot in the slide shows a restricted query being performed. In Enter-Query mode, the user enters :i in the ID field and :n in the Last Name field. When the user clicks Execute Query, the Query/Where dialog box appears, where the user enters the WHERE clause :i=104 OR n BETWEEN 'F' and 'H'. This query returns customer 104 along with all customers whose last name starts with F through G (although in an Oracle database the SQL operator BETWEEN is inclusive, any last name beginning with “H” would be greater than the single letter “H” and so would not be included.)

Example

- To restrict the query to orders with a Sales_Rep_Id (:S) of 11 or an Order_Id (:O) between 100 and 200, enter the following in the Query/Where dialog box:
:S = 11 OR :O between 100 and 200
- To sort the data by Sales_Rep_Id (:S), enter the following in the Query/Where dialog box:
ORDER BY :S

Inserting, Updating, and Deleting Records



ORACLE

2 - 26

Copyright © 2009, Oracle. All rights reserved.

Inserting, Updating, and Deleting Records

Upon entering a typical form module, you are in Normal mode. This means that Forms Builder regards anything that you enter into a blank record as an insert and anything that you enter over an existing record as an update. The graphics in the slide depict the fact that all deletes, updates, and inserts that are performed in Forms are retained in memory until they are committed to the database.

How to Insert a Record

To insert a record, perform the following steps:

1. Ensure that you have the cursor positioned on a blank record by performing one of the following steps:
 - Scroll down until you find a blank record (always the last in the block).
 - Select Record > Insert.
 - Click Insert Record (green +).
 - Press the appropriate function key.
2. Enter the data into the relevant items.

Inserting, Updating, and Deleting Records (continued)

How to Update a Record

To update a record, perform the following steps:

1. Select Query > Enter.
2. Enter the search criteria to retrieve the appropriate record.
3. Select Query > Execute to retrieve all the records that satisfy your specific search criteria.
4. Scroll through the records, and stop at the record that is to be updated.
5. Update the record.

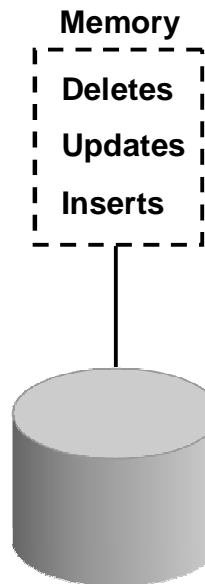
How to Delete a Record

To delete a record, perform the following steps:

1. Select Query > Enter.
2. Enter the search criteria to retrieve the appropriate record.
3. Select Query > Execute to retrieve all records that satisfy your specific search criteria.
4. Scroll through the records, and stop at the record that is to be deleted. Delete the record by performing one of the following actions:
 - Select Record > Remove to clear the record and mark it for deletion.
 - Click Remove Record (red X) to clear the record and mark it for deletion.
 - Press the appropriate function key.

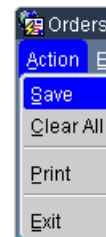
Making Changes Permanent

- Select Action > Save to make changes permanent.
- Select Action > Clear All to discard changes.



To commit or roll back:

Menu



or
Toolbar



Making Changes Permanent

As previously stated and as depicted by the slide graphics, changes are retained in memory. To make any inserts, updates, or deletes permanent, you must save (commit) them to the database. To do this, perform one of the following actions:

- Select Action > Save (shown in the top right screenshot).
- Click Save on the Menu toolbar (shown in the bottom right screenshot).

Discarding Inserts, Updates, and Deletes

To discard any inserts, updates, or deletes, you must clear the records (roll back) instead of saving. Perform a rollback by selecting Action > Clear All.

Exiting a Run-Time Session

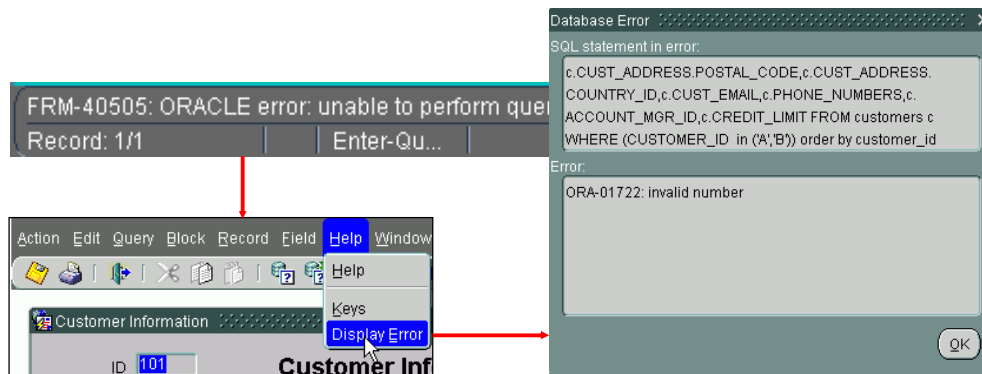
You exit the run-time session by performing one of the following actions:

- Select Action > Exit.
- Click Exit.
- Press the appropriate function key.

Note: By default, you cannot exit the form while you have unsaved updates, inserts, or deletes. You need to either save or undo the changes before you can exit.

Displaying Errors

- Use to view Oracle errors.
- Select Help > Display Error.
- Shows the Database Error window:
 - SQL statement
 - Error information



ORACLE

2 - 29

Copyright © 2009, Oracle. All rights reserved.

Displaying Errors

If an Oracle error is displayed on the message line while you are operating a Forms application, you can view the underlying SQL code by selecting Help > Display Error.

The screenshots in the slide show, at the top left, a form's status line showing error FRM-40404: ORACLE error: unable to perform query. The bottom left screenshot shows selecting Help > Display Error from the menu, and the screenshot at the right shows the Database Error window that shows the SQL query statement at the top and the database error at the bottom: ORA-01722: invalid number.

Example

The following is the SQL statement in error and its corresponding error:

```
SELECT order_id, order_date, order_mode, order_status,
       customer_id, sales_rep_id
FROM orders
WHERE (order_id in ('a','b'))
ORA-01722: invalid number
```

Note: Selecting Help > Display Error displays only those errors where the error on the message line is preceded by ORACLE error.

Summary

In this lesson, you should have learned that:

- You can use WebLogic Server on the development machine to run a Forms application in a Web browser
- At run time:
 - The Forms Client is downloaded
 - The Forms Servlet creates a start HTML file
 - The Forms Listener Servlet starts a run-time session and maintains communication between it and the Forms Client
 - The Forms Runtime Engine carries out application logic and maintains a database connection on behalf of the Forms Client

ORACLE

2 - 30

Copyright © 2009, Oracle. All rights reserved.

Summary

This lesson introduced the operator interface of Forms Builder. The following concepts were covered in this lesson:

- Starting WebLogic Server on the development machine
- The run-time environment for Forms:
 - Running a form in a browser
 - Starting a run-time session: The Forms Servlet, the Forms Client, and the Forms Listener Servlet
- The data elements of a form

Summary

- When you run a form, you see a Java applet running in a browser and displaying a menu, menu toolbar, console, and several kinds of data elements
- Users navigate a Forms application using the menu, toolbar, the mouse, buttons, or function keys
- The two main modes of operation are Normal mode and Enter-Query mode
- Executing a query returns all records, unless the query is restricted by search criteria

ORACLE

2 - 31

Copyright © 2009, Oracle. All rights reserved.

Summary (continued)

- Elements of a running form
- Navigation methods
- Modes of operation:
 - Normal mode
 - Enter-Query mode(There is also a Query mode that occurs when the form is fetching records; the operator cannot interact with the form in Query mode.)
- Retrieving data by performing:
 - Restricted queries: You supply the search criteria.
 - Unrestricted queries: You supply no search criteria.

Summary

- In Normal mode, you can insert, update, and delete records and commit changes to the database
- You display database errors from the menu (Help > Display Error)

ORACLE

2 - 32

Copyright © 2009, Oracle. All rights reserved.

Summary (continued)

- Inserting, updating, and deleting records
- Saving or clearing changes
- Displaying database errors

Practice 2: Overview

This practice covers the following topics:

- Starting the Forms WebLogic Managed Server
- Running the course application:
 - Querying records
 - Inserting a record
 - Updating a record
 - Deleting a record

ORACLE

2 - 33

Copyright © 2009, Oracle. All rights reserved.

Practice 2: Overview

In this practice session, you start an instance of the Forms managed WebLogic Server to function as a Web server on your local machine. You run the course application in a browser and execute both unrestricted and restricted queries. You navigate through the application and use it to insert, update, and delete records.

3

Working in the Forms Environment

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Identify the main Forms executables
- Describe the Forms module types
- Describe Forms Builder components
- Navigate the Forms Builder interface
- Identify the main objects in a form module
- Customize the Forms Builder session
- Use the online Help
- Set environment variables for design and run time
- Run a form from within Forms Builder

ORACLE

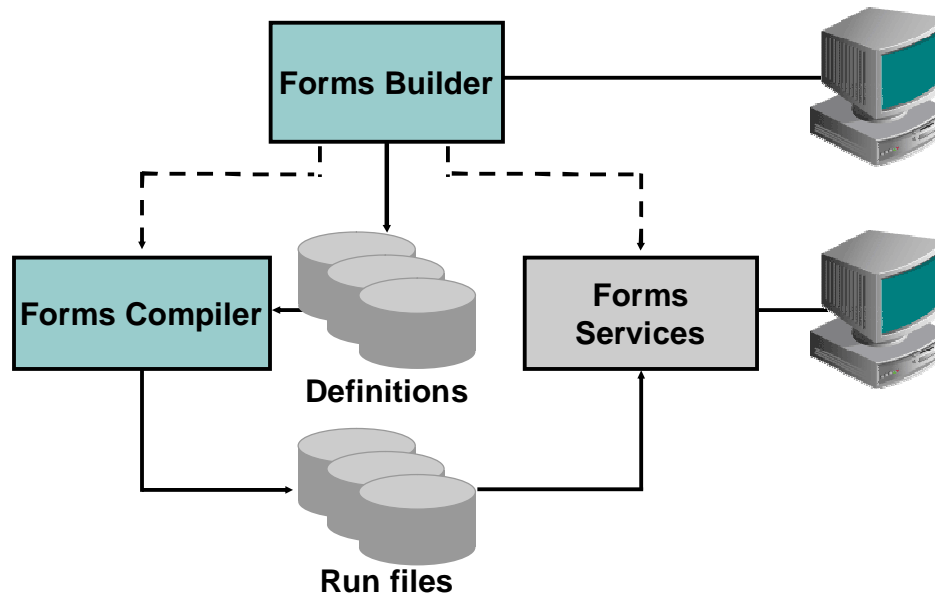
3 - 2

Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

This lesson provides you with an overview of Forms Builder, including a high-level description of its components and object hierarchy. Using this knowledge, you can plan and implement the structure of your Forms applications.

Forms Developer Executables



Forms Developer Executables

Forms Builder includes the following two executables (components) that you can access as the designer of applications.

- **Forms Builder:** This is the application-building component of Oracle Forms Developer. You can use Forms Builder to design and store the definitions of form, menu, and library documents. While in the Forms Builder, you can invoke the other component, Forms Compiler. You must run the Forms Builder component in a graphical user interface (GUI) environment in order to use its graphical design facilities.
- **Forms Compiler:** After your form is built, use the Forms Compiler. This reads the definition of your module and creates an executable run file.

Invoking Forms Builder Executables

In a GUI environment, you usually store commands to invoke Forms Builder components in menus and icons for convenient access. You can also enter these commands on the command line. For example:

```
FRMBLD [my_form] [scott/tiger@my_database]
```

Note: Commands for invoking the product components vary according to platform.

Forms Developer Executables (continued)

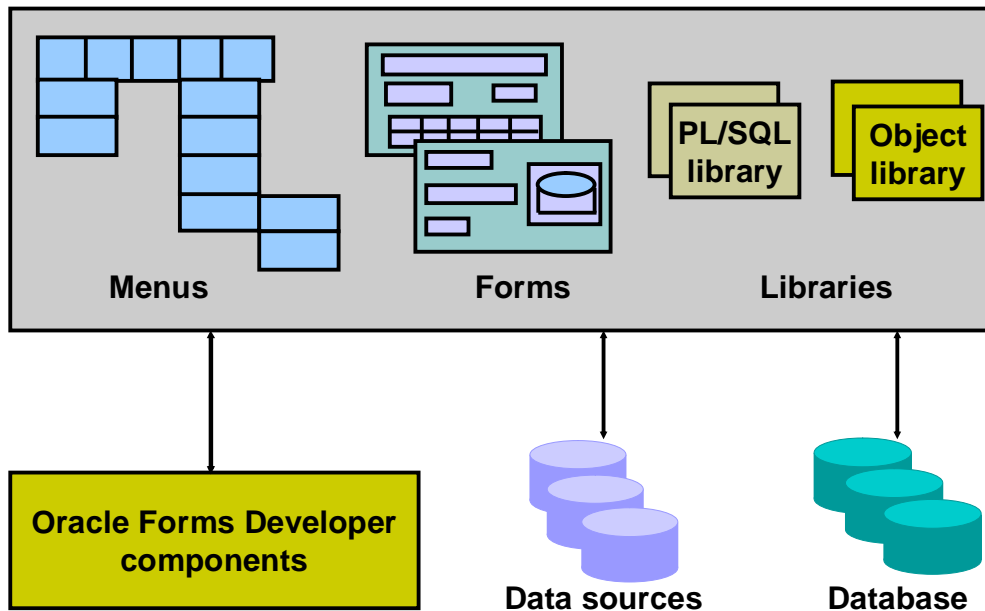
Forms Services

Because Forms applications are Web based, it is not possible to run them directly from the command line. Instead, they are invoked by entering a URL, directed to Forms Services, in a browser.

The files used at run time must already have been compiled by the Forms Compiler component. These files must reside on the middle-tier machine in a directory accessible to the Forms Runtime Engine (in `FORMS_PATH`).

To test your applications, you can also access Forms Services directly from Forms Builder by setting certain preferences, as described later in this lesson.

Forms Developer Module Types



3 - 5

Copyright © 2009, Oracle. All rights reserved.

ORACLE

Forms Developer Module Types

A Forms application can consist of many modules—that is, files. A module is a major component of your application and is the basis for storage and ownership. A module owns the objects that it contains.

A Forms Developer module can be of the following types, as pictured in the graphics in the slide:

- **Form:** As the main component of an application, the form module presents the objects and data that users can see or interact with. Data items in a form are arranged into records.
- **Menu:** A menu module can consist of a hierarchy of menus, each with selectable items. Forms Builder provides the default menu for every form. The default menu includes commands for all basic database operations, such as insert, delete, query, and so on. If your application has specific requirements that are not met by the default menu, you can create a custom menu module. You can use a menu module with multiple forms.

Forms Developer Module Types (continued)

- **PL/SQL Library:** A PL/SQL library is a collection of PL/SQL program units whose code can be referenced and called from other modules. PL/SQL library documents can contain program units that can be used by other form and menu modules.
- **Object Library:** An object library is a collection of form objects that you can use in other modules. You can create it to store, maintain, and distribute standard objects that can be reused across the entire development organization.

You can build an application from multiple form modules, menu modules, and library documents as needed. A form module can be run independently, but menu modules, PL/SQL libraries, and object libraries are functional only when attached to or included in a form module.

Forms Builder: Key Features

With Forms Builder you can:

- Provide an interface for users to insert, update, delete, and query data
- Present data as text, image, and custom controls
- Control forms across several windows and database transactions
- Use integrated menus
- Send data to Oracle Reports

ORACLE

3 - 7

Copyright © 2009, Oracle. All rights reserved.

Forms Builder: Key Features

Forms Builder is a major component of Oracle Forms Developer. You can use Forms Builder to quickly develop form-based applications for presenting and manipulating data in a variety of ways.

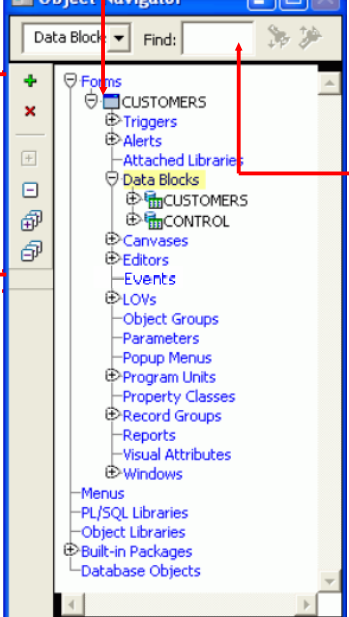
Users of Forms Builder applications can:

- Insert, update, delete, and query data by using a variety of interface items
- Present data by using text, image, and custom controls, including JavaBeans and Pluggable Java Components
- Control forms across several windows and database transactions
- Access comprehensive facilities by using integrated menus
- Send data directly to Oracle Reports

As the designer of Forms Builder applications, you can:

- Design forms that use a number of data sources, including Oracle databases
- Build applications quickly and easily by using powerful GUI development tools
- Design applications for Internet deployment
- Copy and move objects and their properties easily between applications
- Use design features such as wizards, the Layout Editor, Object Navigator, and PL/SQL Editor

Forms Builder Components: Object Navigator

- Objects displayed hierarchically
 - Toolbar to create, delete or unload, and expand or contract
- 
- Icons to represent objects
 - Fast search feature

ORACLE

3 - 8

Copyright © 2009, Oracle. All rights reserved.

Forms Builder Components: Object Navigator

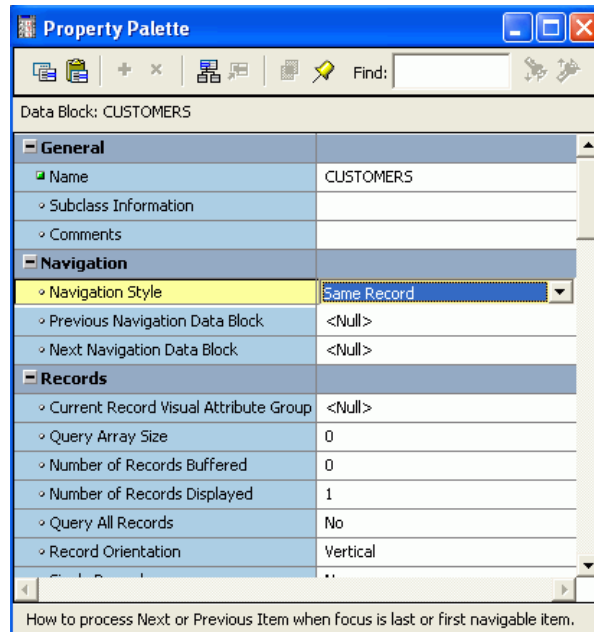
The interface components of the Forms Builder tool help to provide the flexibility and productivity of the Oracle Forms development environment.

The Object Navigator, pictured in the screenshot in the slide, is a hierarchical browsing and editing interface. You can use the Object Navigator to locate and manipulate application objects quickly and easily. Features include:

- A hierarchy represented by indentation and expandable nodes (Top-level nodes show application objects [forms, menus, and libraries], database objects, and built-in packages. All other nodes and the objects they contain are indented to indicate that they belong to these higher-level nodes.)
- Find field and icons, enabling forward and backward searches for any level of node or for an individual item in a node
- Icons in the vertical toolbar to create, delete or unload, and expand or contract
- An icon next to each object to indicate the object type

Forms Builder Components: Property Palette

- Copy and paste properties
- Fast search feature



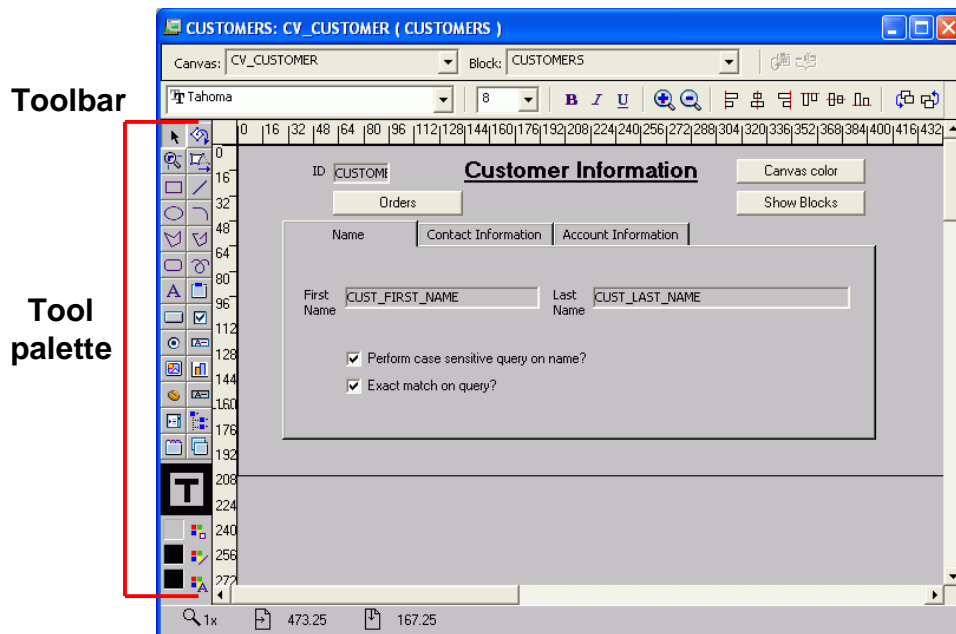
ORACLE

Forms Builder Components: Property Palette

All objects in a module, including the module itself, have properties that you can see and modify in the Property Palette, shown in the screenshot in the slide. Features include:

- Copy and reuse properties from another object
- Find field and icons, similar to the Object Navigator

Forms Builder Components: Layout Editor



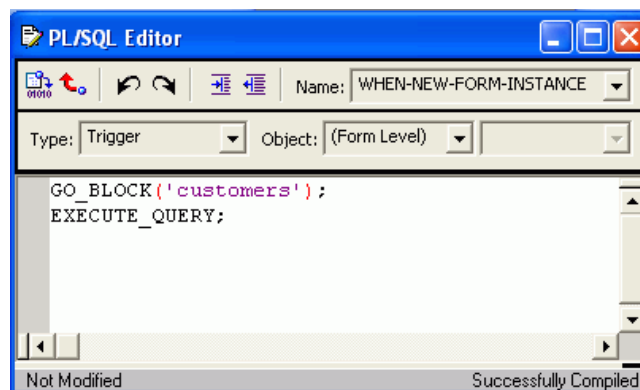
Forms Builder Components: Layout Editor

The Layout Editor, shown in the screenshot, is a graphical design facility for creating and arranging interface items and graphical objects in your application. You can use the tool palette and the toolbar available in the Layout Editor to design the style, color, size, and arrangement of visual objects in the application. The layout can include graphical objects and images.

Forms Builder Components: PL/SQL Editor

With the PL/SQL Editor, you can:

- Use PL/SQL in Forms
- Enter and compile code



Forms Builder Components: PL/SQL Editor

The PL/SQL Editor, shown in the screenshot, enables you to incorporate PL/SQL code objects into your form. Code objects in Forms Developer include event triggers, subprograms (functions and procedures), menu item commands, menu startup code, and packages. You enter and compile code in the PL/SQL Editor. You will learn more about the PL/SQL Editor in later lessons when you use it to code triggers in Forms Builder.

Getting Started in the Forms Builder Interface



- Start Forms Builder.
- Connect to the database:
 - Menu:
Select File > Connect.
 - Or
 - Toolbar:
Click Connect.

ORACLE

3 - 12

Copyright © 2009, Oracle. All rights reserved.

Getting Started in the Forms Builder Interface

Starting Forms Builder

To start Forms Builder on Windows, invoke it from the Start menu: Programs > Oracle Classic Instance > Developer Tools > Forms Builder.

When you invoke Forms Builder, a Welcome dialog box appears. If you click Cancel to dismiss the dialog box, you see the Object Navigator and an empty new module.

If you build applications that access database objects, you must connect to a database account from the Forms Builder. Connect to a database if you need to:

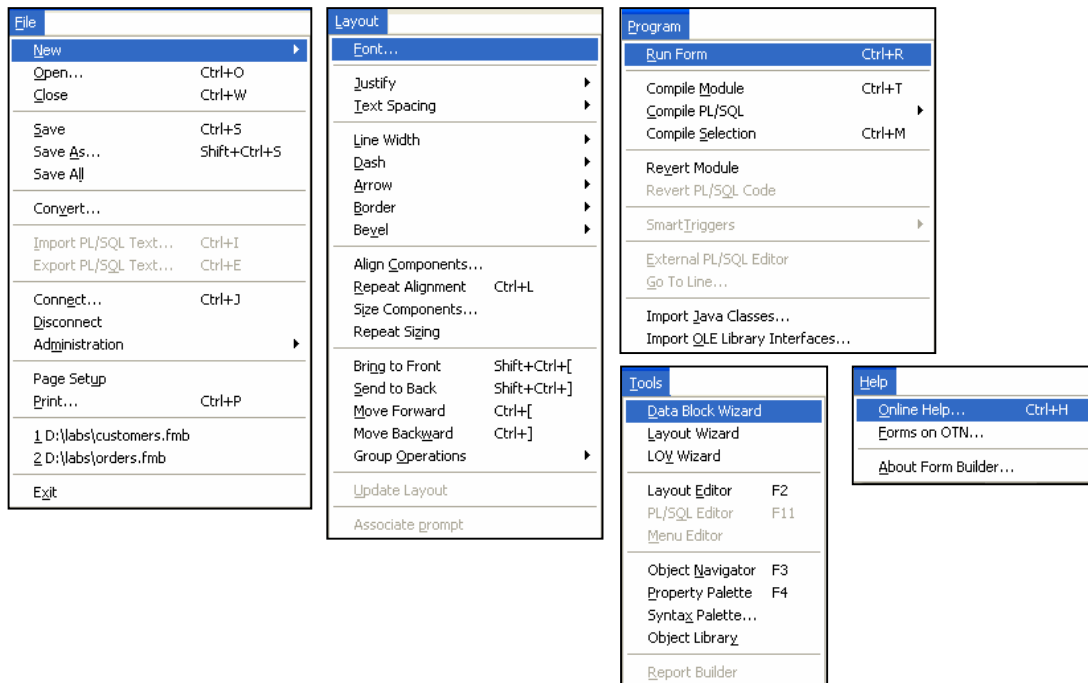
- Compile code that contains SQL
- Access database objects in the Object Navigator
- Create Oracle Forms Developer objects that are based on database objects

Connecting to a Database

1. Select File > Connect from the menu, or click the Connect icon on the toolbar as shown in the slide.
2. Enter the database user and password in the Connect dialog box. If not connecting to the default database, provide the necessary connect string or database alias.

Note: Oracle Forms Builder automatically displays the Connect dialog box if you try to perform a task that requires connection.

Navigating the Forms Builder Main Menu



Navigating the Forms Builder Main Menu

The Forms Builder main menu contains options that enable you to create, modify, and manage your form modules.

Common Menu Features

The following table describes some common features in GUI menus:

Feature	Description
Underline	Shortcut key: Alt + letter
Ellipsis (...)	Additional input, usually by using a dialog box
▶	Menu option has a submenu
Windows menu	List of open windows; select any window to make it active
Help	List of help facilities and an About box

Native GUI Interface

The menus shown in the slide are screenshots of the Windows environment. However, menus appear with the same look and feel of your native GUI interface.

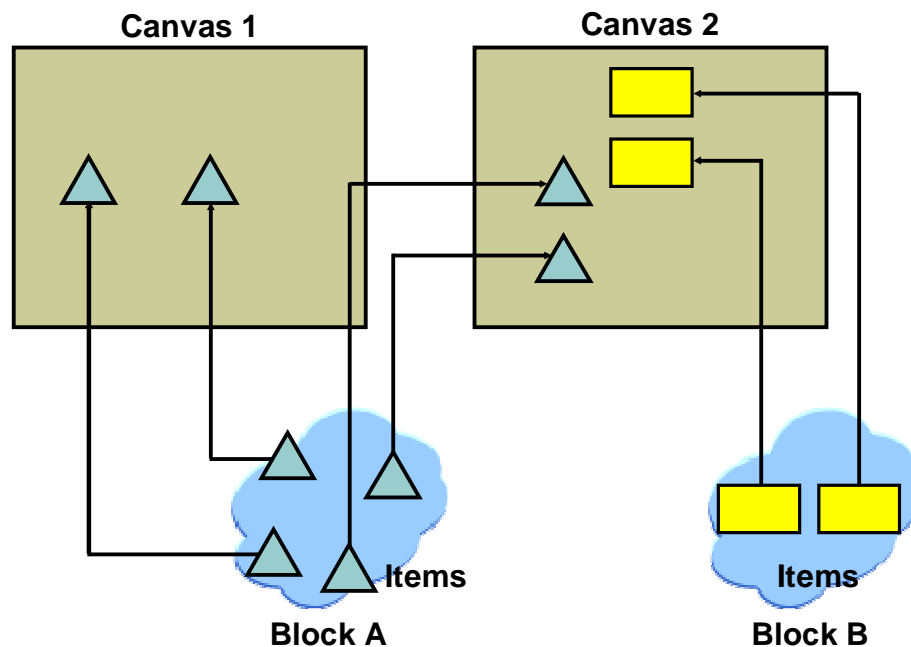
For example, in Motif, the Windows Print Dialog options appear as submenus of the Font menu.

Navigating the Forms Builder Main Menu (continued)

Forms Builder Main Menu

Menu Item	Description
File *	Common file utilities, such as open, save, connect, and administration
Edit	Cut, copy, paste, create, preferences, and so on
View	Switch view in current window; options vary depending on context
Layout *	Common commands for use in the Layout Editor
Program *	Includes compilation and commands related to code
Debug	Invokes debugger functionality
Tools *	Access to wizards and other Forms Builder components
Window	Access to windows displayed in Forms Builder
Help *	Access to built-in and online help systems
* Pictured in the slide	

Items, Blocks, and Canvases: Overview



ORACLE

3 - 15

Copyright © 2009, Oracle. All rights reserved.

Items, Blocks, and Canvases: Overview

Forms modules make up the main “body” of an Oracle Forms Developer application. They can consist of many types of objects, some of which are visible to the user at run time.

The three major objects in a form, as shown in the graphics in the slide, are:

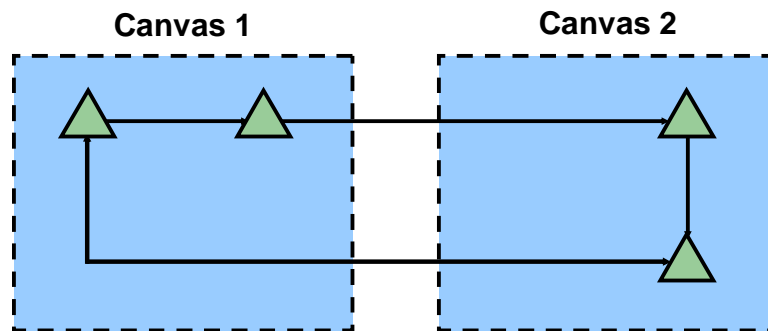
- **Items:** These are interface objects that present data values to the user or enable the user to interact with the form, depending upon the item type. There are several types of items. Items are logically grouped into blocks and visibly arranged on canvases.
- **Blocks:** A block is the intermediate building unit for forms. Each form consists of one or more blocks. A block is the logical owner of items, and each item in a form belongs to a block. Items in one block are logically related (for example, they may correspond to columns in the same database table or may need to be part of the same navigation cycle). Blocks, therefore, provide a mechanism for grouping related items into a functional unit for storing, displaying, and manipulating records.

Items, Blocks, and Canvases: Overview (continued)

- **Canvases:** A canvas is a “surface” where visual objects, such as graphics and items, are arranged. A form module can have several canvases (such as the pages of a paper form). A canvas can display items from one or more blocks. To see a canvas and its items, you must display the canvas in a window. A window can have more than one canvas. By default, all canvases in a form appear in the same window (which could mean you see only one canvas at a time), but you can assign separate windows for each canvas so that several canvases can be viewed at once. You learn more about windows and canvases in the lessons titled “Creating Windows and Content Canvases” and “Working with Other Canvas Types.”

Note: Items in one block do not need to be physically grouped. They can span many canvases (and windows). A canvas is similar to a picture portrait, and a window is similar to a picture frame. Just as you need a picture frame to display a picture portrait, you need a window to display a canvas and its contents.

Navigating in a Forms Module



ORACLE

3 - 17

Copyright © 2009, Oracle. All rights reserved.

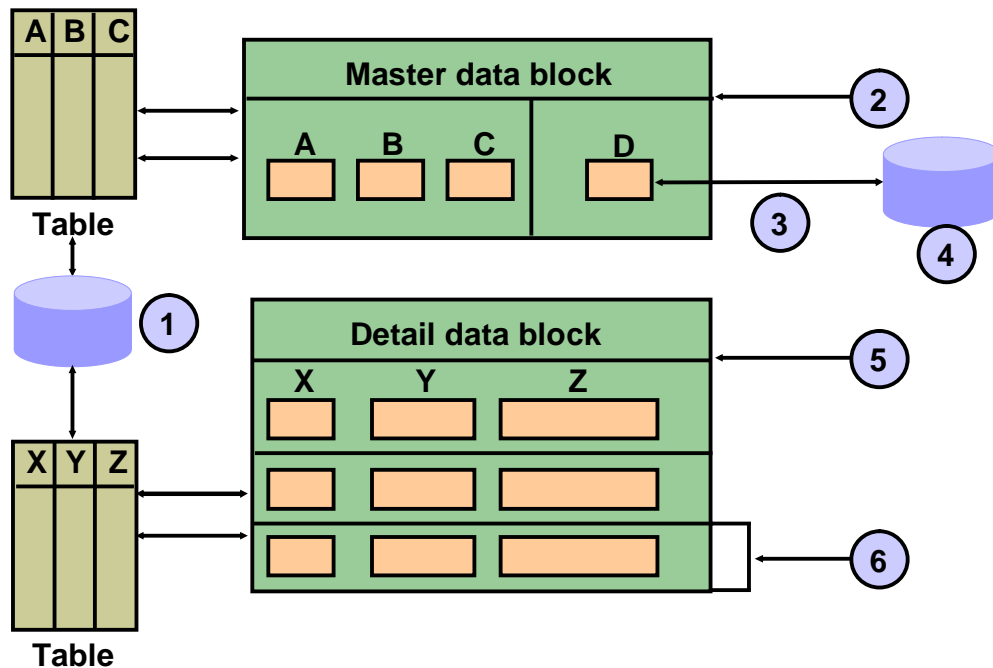
Navigating in a Forms Module

When you run a form, you principally navigate by way of items and blocks, not by canvases. Each item has a sequenced position within its block, and each block has a sequenced position in the form.

When a user requests to move to the next item in a block, focus will be set on the next item in sequence, wherever that may be. Default navigation follows the order of items in a block as listed in the Object Navigator, but you can change this. If the next item is on a different canvas, as shown in the slide, Forms displays that canvas automatically. Similarly, users can request to move to the next block (or previous block). If the first item in this block resides on another canvas, that canvas is displayed automatically.

If you can already see the item that you want to move to, you may click it. You can also program mechanisms into the application to enable navigation in other ways. You learn more about this in the lesson titled “Navigation.”

Using Blocks



ORACLE

3 - 18

Copyright © 2009, Oracle. All rights reserved.

Using Blocks

In Forms Builder, there are two main types of blocks: data blocks and control blocks.

Data Blocks

When you build database applications with Forms Builder, many of the blocks will be data blocks. A data block is associated with a specific database table (or view), a stored procedure, a FROM clause query, or transactional triggers.

If it is based on a table (or view), the data block can be based on only one base table, even though the data block can be programmed to access data from more than one table and data sources. By default, the association between a data block and the database enables the user to automatically access and manipulate data in the database. However, to access data from other tables (nonbase tables), you need to write triggers.

The graphics in the slide depict the following:

1. Base table source
2. Single-record data block
3. Trigger access
4. Nonbase table source
5. Multirecord data block
6. Record

Using Blocks (continued)

For a base table, Forms Builder can automatically perform the following actions:

- Create items in the data block to correspond to columns in the table. (These items are data items or base-table items.)
- Produce code in the form to employ the rules of the table's constraints.
- Generate SQL at run time (implicit SQL) to insert, update, delete, and query rows in the base table, based on the user's actions.

At run time, you can use standard function keys, buttons, menu options, or standard toolbar options to initiate query, insert, update, or delete operations on base tables, and the subsequent commit of the transaction.

Control Blocks

A control block is not associated with a database, and its items do not relate to any columns within any database table. Its items are called control items. For example, you can create many buttons in your module to initiate certain actions, and you can logically group these buttons in a control block.

Master Versus Detail Blocks

To support the relationship between data blocks and their underlying base tables, you can define one data block as the detail (child) of a master (parent) data block. This links primary key and foreign key values across data blocks, and synchronizes the data that these data blocks display.

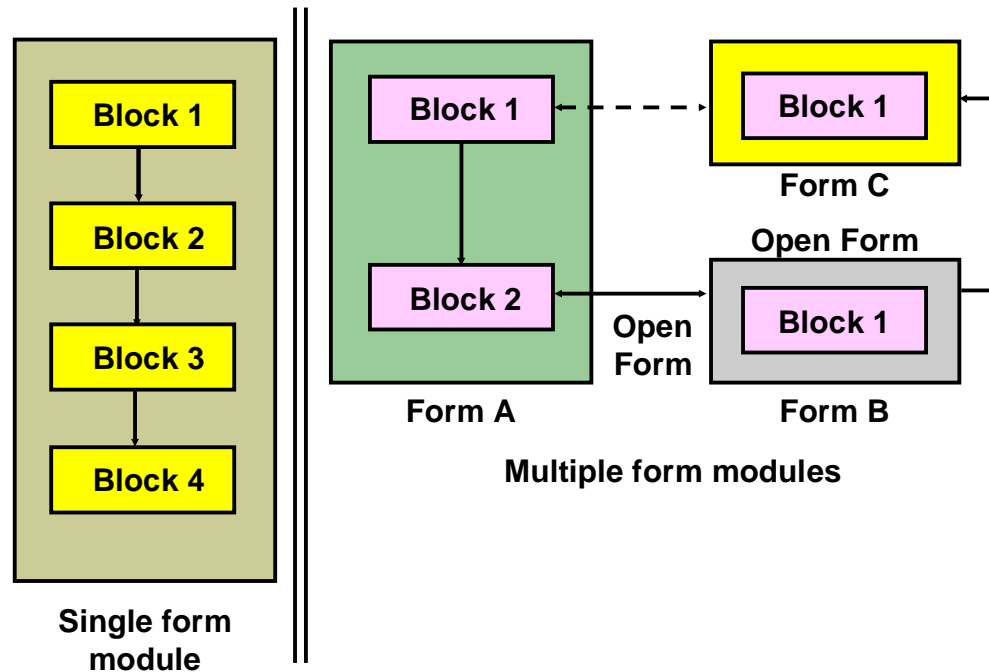
Forms Builder automatically generates the objects and code needed to support master-detail relationships. As the designer, you need only request it.

Note: If your application requires it, you can also create independent data blocks in which there is no relationship between the two data blocks.

Single-Record Versus Multirecord Blocks

You can design a data block to show one record at a time (single-record block) or several records at once (multirecord block). Usually, you create a single-record data block to show master block data and a multirecord data block to show detail block data. In either case, records in a data block that are currently not visible on the screen are stored in a block buffer.

Designing Multiblock and Multiform Applications



Designing Multiblock and Multiform Applications

Typically, a Forms Builder application consists of more than one data block. With more than one data block, you can do the following:

- Separate the navigation cycle of one group of items from another.
- Map each data block to a different database table. (You can have one base table per data block.)
- Produce a master-detail form, with a master data block and corresponding detail data blocks that are related to the master.

You can create a large form module with many data blocks, similar to the one shown in the left side of the slide. Alternatively, you can create several smaller form modules with fewer data blocks in each, as shown by the graphic in the right side of the slide.

Generally, a modular application with several smaller form modules has the following characteristics:

- Modules are loaded only when their components are required, thus conserving memory.
- Maintenance can occur on one module without regenerating or loading the others.
- Forms can call upon one another, as required.

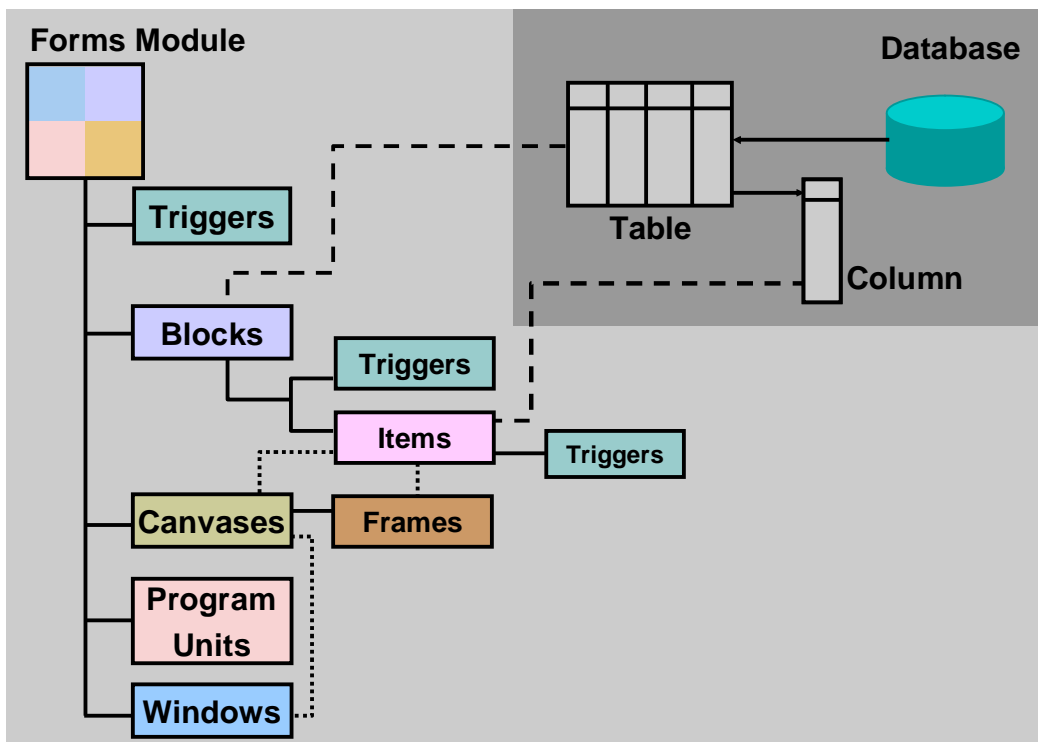
Multiblock and Multiform Applications (continued)

- Parallel development can be carried out by different team members on different components.

Here are some points to consider when grouping data blocks in the application:

Data Blocks in the Same Form Module	Data Blocks in Different Form Modules
The data blocks can be directly linked in master-detail relationships.	The data blocks cannot be linked by the standard interblock relations.
Navigation between data blocks is handled by default functionality.	Navigation between data blocks of different forms is programmed by the designer (although mouse navigation to visible items can be automatic).

Form Module Hierarchy



ORACLE

3 - 22

Copyright © 2009, Oracle. All rights reserved.

Form Module Hierarchy

The hierarchy of a form module is depicted by the graphics in the slide that show some of the objects of a module.

Triggers: Using triggers, you can write PL/SQL code to add functionality to your form. Triggers can be written at form, block, or item level in a Forms module.

Blocks: A form module is made up of one or more blocks. A data block is based on a database object, such as a table or a view. A data block can contain both data items, which represent columns in the base table, and control items. The dashed lines in the slide represent the relationships between database objects (table and column) and Forms objects (block and item.)

Canvases: To be visible to the end user, each item in a block must appear on a canvas. A frame can be created to arrange data block items on the canvas.

Program Units: User-named program units enable you to write additional PL/SQL code through procedures, functions, and packages.

Windows: To be visible to the end user, each canvas must appear in a window. A form module can have one or more canvases and windows.

Form Module Hierarchy (continued)

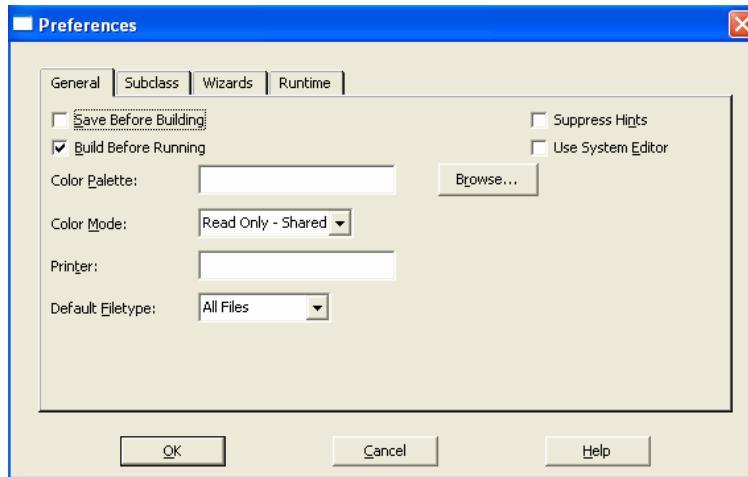
Object Hierarchy

You can create many types of objects in a form module. They are discussed in more detail in later lessons.

In the following table, note that some objects are associated, even though one might not be “owned” by the other. These associations are shown in the slide by the dotted lines connecting items with canvases and frames and also connecting canvases with windows.

Object	Description
Block	Logical section of a form; owned by the Forms module
Item	Member of a data block (items are functionally grouped into records that can represent a row in a table, but records are not Forms objects)
Canvas	The surface where visual objects are arranged; owned by the form module; can contain text and graphics—static information that the user cannot interact with—as well as items
Window	Produced to contain the views of canvases; owned by the form module
Frame	A graphic object that appears on a canvas, is owned by that canvas, and is used to arrange the items within a data block, although items can appear on a canvas without a frame object
User-named program unit	Named procedure, function, or package owned by the form module
Trigger	PL/SQL block executed on an event; depending on scope, can be owned by the form module, a data block, or an item
Other objects	Mainly owned by the form module itself; include alerts, attached libraries, editors, events, object groups, parameters, property classes, record groups, report objects, and visual attributes

Customizing Your Forms Builder Session



Customizing Your Forms Builder Session

You can use preferences to customize some aspects of your Forms Builder session. There are four tabs in the Preferences dialog box. The screenshot shows the General tab and the preferences that you can set there.

To see a description of each preference, click Help in the Preferences dialog box or press the Help key (F1 for Windows).

In addition to session preferences, you can also set run-time preferences that apply to running your form from within the builder.

To modify preferences, perform the following steps:

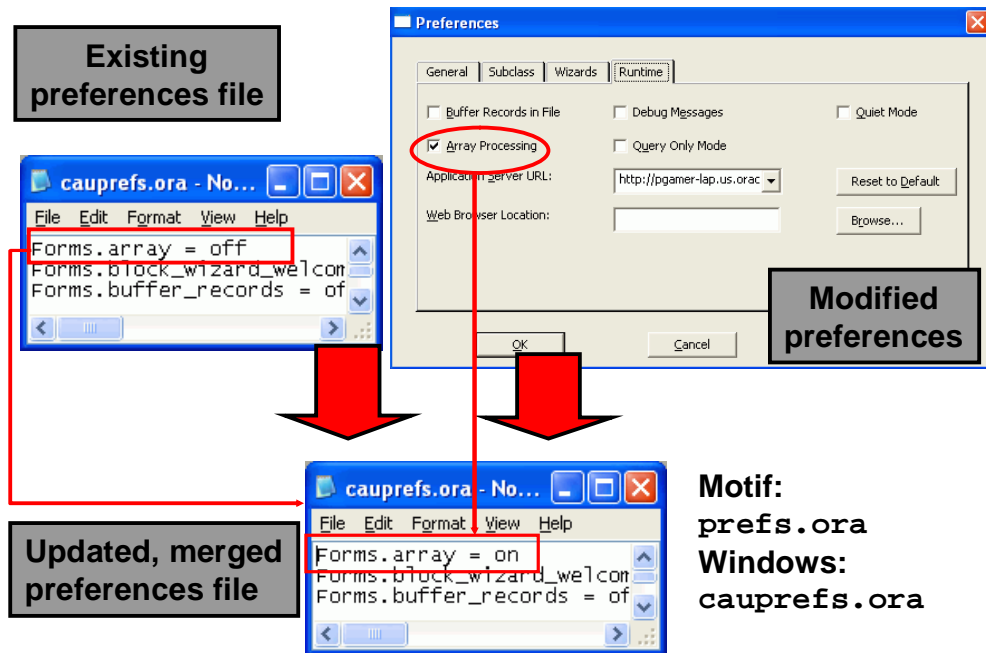
1. Select Edit > Preferences.
2. Specify any options that you require.
3. Click OK to save the changes or Cancel to cancel the changes.

Customizing Your Forms Builder Session (continued)

You can specify several related preferences on each of the Preference tabs. The following table describes one preference on each of the tabs:

Tab	Preference Name	Description
General	Build Before Running	Determines whether Forms Builder automatically compiles the active module when you run a form. This option enables you to avoid issuing separate Compile and Run commands each time you modify and run a form.
Subclass	Subclassing Path	Options for keeping or removing the subclassing path
Wizards	Welcome Dialog	Check box to suppress or display the first Welcome dialog box. There are several similar check boxes.
Runtime	Array Processing	Determines whether Forms Builder processes groups of records at a time, reducing network traffic and increasing performance

Saving Preferences



3 - 26

Copyright © 2009, Oracle. All rights reserved.

Saving Preferences

When you click OK in the Preferences dialog box, Oracle Forms Builder updates your current session with the changes.

When you exit the builder (File > Exit), Oracle Forms Builder writes the changes to a preference file for future sessions. The name of the preference file varies on different platforms.

Oracle Forms Developer and Oracle Reports Developer share the same preference file. If the preference file already exists, Oracle Forms Developer merges its changes with the existing file. This does not affect preferences for Reports.

Each option in the preference file is prefixed by the tool name to which it belongs.

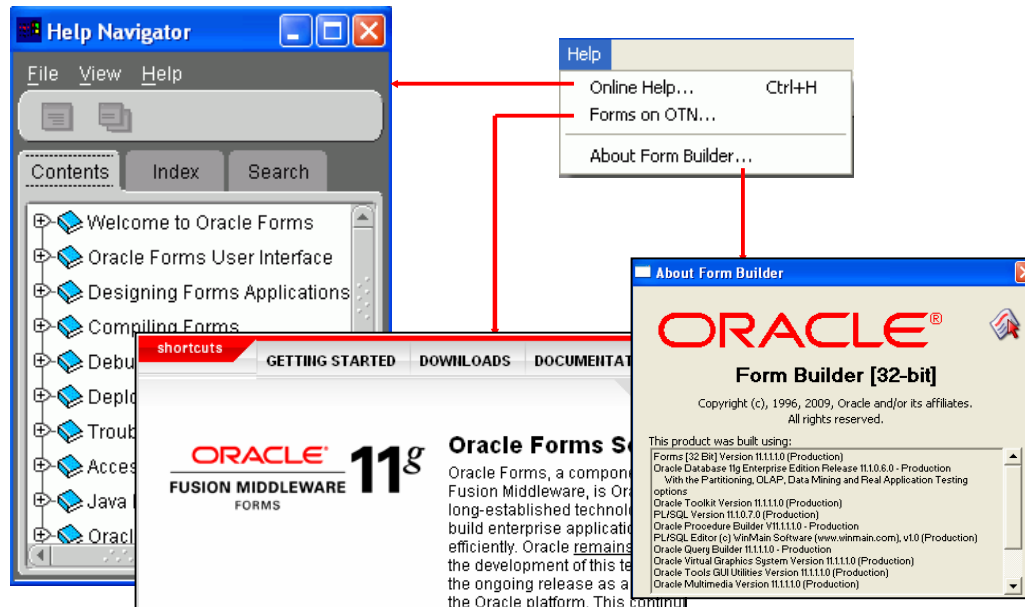
Example:

```
Forms.build_before_run = on
Forms.welcome_dialog = off
```

Oracle Forms Developer reads the preference file whenever you invoke Forms Builder. Oracle Reports Developer reads the preference file whenever you invoke Reports Builder.

Note: The preferences file is an editable text file. If possible, however, you should use the Preferences dialog box to alter the options. The graphics in the slide show how to change a preference in the dialog box and the effect it has on the `cauprefs.ora` file.

Using the Online Help System



3 - 27

Copyright © 2009, Oracle. All rights reserved.

Using the Online Help System

You can invoke online Help from the menu, as shown in the slide. The following table describes the Help menu options in Forms Builder:

Help Menu Option	Description
Online Help	Comprehensive online Help window with three tabs. The Contents tab, shown in the screenshot at the left of the slide, provides access to a variety of manuals and references. There are also Index and Search tabs. The Help key (F1 for Windows NT/95) displays context-sensitive online Help at any place in the Builder.
Forms on OTN	The latest product information on Oracle Technology Network, shown by the middle screenshot in the slide
About Form Builder	Shown at the bottom right of the slide, this is a separate window that shows product components and their version numbers. When you are connected to a database server, it also displays similar information for server-side product components.

You can also invoke context-sensitive online Help from Forms Builder by pressing the Help key (F1 on Windows).

Defining Forms Environment Variables for Design Time

Set on the Forms Builder machine:

- FORMS_BUILDER_CLASSPATH
- UI_ICON
- UI_ICON_EXTENSION
- FORMS_HIDE_OBR_PARAMS

Windows: Modify in Registry
(REGEDIT.EXE or
REGEDT32.EXE)

ORACLE

3 - 28

Copyright © 2009, Oracle. All rights reserved.

Defining Forms Environment Variables for Design Time

Design time environment variables help govern the behavior of Forms Builder. These environment variables must be set on the machine where Forms Builder is installed. For example, on Windows, you set them in the Windows Registry.

Java class files: Forms Builder needs access to certain Java classes for some of its features, such as Help, the debugger, and the Java importer. You set FORMS_BUILDER_CLASSPATH so that Forms Builder can find the Java classes that it needs during the development and testing of an application.

Icon files: Forms Builder enables you to create buttons with iconic images. You can use .ico, .gif, or .jpg images. You set UI_ICON to the directory path where these images are located. You set the UI_ICON_EXTENSION environment variable to indicate to Forms Builder which type of image to display on buttons. Valid values are ico (the default), gif, or jpg.

Although you can use .ico images to display on iconic buttons within Forms Builder, such images will not be displayed when running the form—use .gif or .jpg files for deployed icons.

Defining Forms Environment Variables for Design Time (continued)

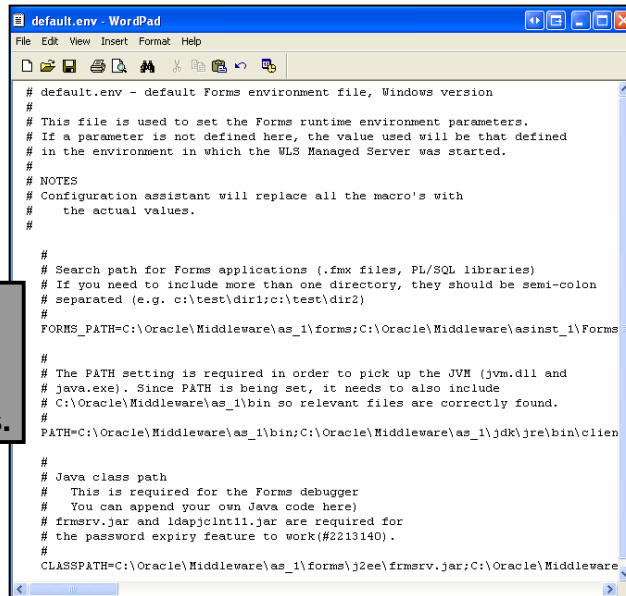
URL parameters: By default, when you run a form from Forms Builder, the parameters that are passed in the URL are hidden. For testing purposes, if you want to be able to see these parameters, such as the form name, you can set `FORMS_HIDE_OBR_PARAMS` to `FALSE` (the default is `TRUE`). OBR stands for one button run, a term used for running a form from within Forms Builder by clicking the Run Form button.

Defining Forms Environment Variables for Run Time

Set on the middle-tier machine (used at run time):

- FORMS_PATH
- ORACLE_PATH
- CLASSPATH

For Forms deployment, the settings in the environment control file override system settings.



```
# default.env - default Forms environment file, Windows version
#
# This file is used to set the Forms runtime environment parameters.
# If a parameter is not defined here, the value used will be that defined
# in the environment in which the WLS Managed Server was started.
#
# NOTES
# Configuration assistant will replace all the macro's with
# the actual values.
#
# Search path for Forms applications (.fmx files, PL/SQL libraries)
# If you need to include more than one directory, they should be semi-colon
# separated (e.g. c:\test\dir1;c:\test\dir2)
#
FORMS_PATH=C:\Oracle\Middleware\as_1\forms;C:\Oracle\Middleware\asinst_1\Forms
#
# The PATH setting is required in order to pick up the JVM (jvm.dll and
# java.exe). Since PATH is being set, it needs to also include
# C:\Oracle\Middleware\as_1\bin so relevant files are correctly found.
#
PATH=C:\Oracle\Middleware\as_1\bin;C:\Oracle\Middleware\as_1\jdk\jre\bin\client
#
# Java class path
# This is required for the Forms debugger
# You can append your own Java code here)
# frmsrv.jar and ldapjclnt11.jar are required for
# the password expiry feature to work(#2213140).
#
CLASSPATH=C:\Oracle\Middleware\as_1\forms\j2ee\frmsrv.jar;C:\Oracle\Middleware
```

Defining Forms Environment Variables for Run Time

Oracle Forms Developer uses many environment variables. These have default values, all of which you can modify in your own environment for different applications.

Setting Search Paths for Run Time

Forms uses some environment variables set on the middle-tier computer to search at run time for files such as forms, menus, and libraries. This enables you to build applications that are portable across platforms and directory structures by avoiding hard-coded paths in file references.

Forms searches the following paths in order until the required file is found:

- The current working directory
- Directories in FORMS_PATH
- Directories in ORACLE_PATH

Although you could set these variables at the machine level, such as in the Windows Registry, it is preferable to set them in the Forms environment file, which is shown in the slide. Settings in this file override system settings for running a Forms application.

Forms Files to Define Run-Time Environment Variables

- Environment control file:
 - `default.env` or
 - Other file specified in the Forms configuration file
- Forms configuration file:
 - `formsweb.cfg` or other
 - Used to specify:
 - System parameters, such as `envFile`
 - User parameters, such as form and user ID
 - Settings for the Java client
 - Other settings

ORACLE

3 - 31

Copyright © 2009, Oracle. All rights reserved.

Forms Files to Define Run-Time Environment Variables

In a Windows 32-bit environment, you can use the Windows Registry to modify these paths, except for `CLASSPATH`, which is set in the System settings of the Control Panel. You can also override these settings at run time in the file that controls the Forms run-time environment, which is the `default.env` file unless a different file is specified. Using the Forms environment file makes it easier to deploy the application on any platform.

You can specify which file to use as the environment file in the Forms configuration file, `formsweb.cfg`, where you can set system parameters such as the name of the environment control file. You can also set parameters to control which form to run, the user ID, aspects of the Java client and the HTML file that contains the Java applet, and many other settings.

The `default.env` and `formsweb.cfg` files are located in the `\stage\formsapp\11.1.1\formsapp\config` subdirectory of the domain home for the Forms-managed WebLogic server.

Setting Date Formats in the Run-Time Environment

- NLS_DATE_FORMAT
- FORMS_USER_DATE_FORMAT
- FORMS_USER_DATETIME_FORMAT
- FORMS_OUTPUT_DATETIME_FORMAT
- FORMS_OUTPUT_DATETIME_FORMAT
- FORMS_ERROR_DATE_FORMAT
- FORMS_ERROR_DATETIME_FORMAT

ORACLE

3 - 32

Copyright © 2009, Oracle. All rights reserved.

Setting Date Formats in the Run-Time Environment

Dates in Oracle Forms Developer applications can be fetched from the database, entered by the end user, or defined in the application itself.

Date Format Masks

In the lesson titled “Working with Text Items” you learn how to specify a format mask for a date item in your form. In addition to the format masks a developer might explicitly define, Forms Builder uses several of its own internal masks. You can use the following environment variables to specify the values for these internal masks:

- **Database date format mask:** Each database session within a Forms application has a single database date format mask. A default value for this mask is established by the Oracle server’s initialization parameter. You can override this value in each new database session for a particular client by setting the client’s NLS_DATE_FORMAT environment variable.

Setting Date Formats in the Run-Time Environment (continued)

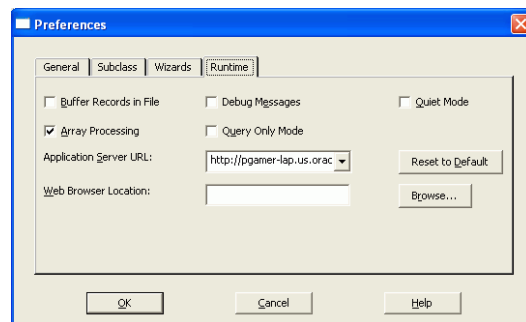
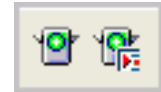
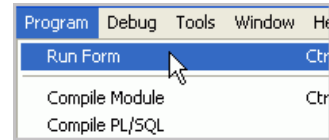
- **Input date format masks:** This mask (potentially, a set of masks) is used to convert a user-entered string into a native format date value. You can set the environment variables `FORMS_USER_DATE_FORMAT` or `FORMS_USER_DATETIME_FORMAT` to specify these format masks.

For example, you can set `FORMS_USER_DATE_FORMAT` to the value `FXFMDD-MM-RRRR`. This forces the user to enter values into date items (with no format mask) in the format exemplified by 30-06-97. The `RRRR` token enables years between 1950 and 2049 to be entered with the century omitted.

- **Output date format masks:** This mask converts dates displayed in output, such as lists of values: `FORMS_OUTPUT_DATE_FORMAT` or `FORMS_OUTPUT_DATETIME_FORMAT`.
- **Error date format masks:** This mask converts dates displayed in error messages: `FORMS_ERROR_DATE_FORMAT` or `FORMS_ERROR_DATETIME_FORMAT`.

Testing a Form with the Run Form Button

- With the Run Form menu command or button, you can:
 - Run a form from Forms Builder
 - Test the form in a three-tier environment
- The Run Form command takes its settings from Preferences:
 - Edit > Preferences
 - Runtime tab
 - Set Web Browser Location if desired.
 - Set Application Server URL to point to Forms Servlet:



<http://127.0.0.1:9001/forms/frmservlet>

ORACLE

Testing a Form with the Run Form Button

The Run Form menu command or button enables you to run a form module in a Web browser from within Forms Builder. This makes it easy to test your application in a three-tier environment, with all components appearing and behaving as they would for a user of the application.

By default, the testing occurs on the development computer where Forms Builder is installed, using the default settings in the Forms Servlet configuration file, with no parameters passed on the URL.

The screenshots show from top to bottom:

- Choosing Run Form from the Program menu
- The Run Form and the Run Form Debug toolbar buttons
- The Runtime tab of the Preferences window, where you set the Application Server URL

Testing a Form with the Run Form Button (continued)

You can modify this three-tier testing environment in the Preferences dialog box by performing the following steps:

1. Click the Runtime tab.
2. Set the Application Server URL: This must be a URL pointing to the Forms Servlet on the middle tier. Note that it is typically on the same machine where you are running Forms Builder. You can also use the `config` parameter to specify a named configuration in the Forms Web configuration file (`formsweb.cfg` by default).

Example for the same machine with the WLS_FORMS managed WebLogic server running on the default port of 9001:

<http://127.0.0.1:9001/forms/frmservlet?config=myapp>

Note: You can enter the default setting in the Application Server URL field by clicking “Reset to Default.”

3. Set the Web Browser Location (needed only if you want to run in a different browser than the default for your machine).

Exiting the Session

The correct way to exit a Forms session is to exit the form (Action > Exit, or click Exit) and then close the browser. If you close the browser without first exiting the form, your session may hang.

You will notice this because you may not be able to recompile the same form, but will receive the error: FRM-30087: Unable to create form file. If this happens, you can open Task Manager and end the `frmweb` process manually.

Summary

In this lesson, you should have learned that:

- The Forms Developer executables are the Forms Builder and the Forms Compiler
- The Forms Developer module types are forms, menus, and libraries
- Forms Builder includes the Object Navigator, the Property Palette, the Layout Editor, and the PL/SQL Editor
- You can use the Object Navigator or the menu and its associated toolbar icons to navigate around the Forms Builder interface

ORACLE

3 - 36

Copyright © 2009, Oracle. All rights reserved.

Summary

- The Forms Developer executables are the Forms Builder and the Forms Compiler
- With Forms Builder, which is an Oracle Forms Developer component, you can develop form-based applications for presenting and manipulating data in a variety of ways. Forms Builder enables screen-based queries, inserts, updates, and deletes of data.
- Forms Builder provides powerful GUI and integration features.
- Applications consist of form modules, menu modules, and library documents.

Summary

- The main objects in a form module are blocks, items, and canvases
- The Edit > Preferences dialog box enables you to customize the Forms Builder session
- The Help menu enables you to use the online Help to look up topics, or you can invoke context-sensitive help
- You can set environment variables in the Forms environment file (for run time) or on the development machine (for design time)
- You can use the Run Form button to run a form from within Forms Builder

ORACLE

3 - 37

Copyright © 2009, Oracle. All rights reserved.

Summary (continued)

- Form modules consist of logical data blocks. A data block is the logical owner of items. Items in one data block do not need to be physically grouped. Items in one data block can span several canvases.
- Environment variables govern the behavior of:
 - Running forms (set in the Forms environment file)
 - Forms Builder (set on the development machine, such as in the Windows Registry for Windows platforms)
- You can run a Forms application from within Forms Builder in order to test it in a browser. You specify the URL to use on the Runtime tab of the Preferences dialog box.

Practice 3: Overview

This practice covers the following topics:

- Setting Forms Builder preferences
- Becoming familiar with the Object Navigator
- Using the Layout Editor to modify the appearance of a form
- Setting environment variables so that the Layout Editor in Forms Builder displays `.gif` images on iconic buttons

ORACLE

3 - 38

Copyright © 2009, Oracle. All rights reserved.

Practice 3: Overview

In this practice, you become familiar with Oracle Forms Builder by performing the following tasks:

- Setting Forms Builder preferences
- Examining the Object Navigator in Forms Builder
- Using the Layout Editor in Forms Builder to modify the appearance of a form
- Setting environment variables so that images display on iconic buttons in the Layout Editor of Forms Builder.

You also run forms from within Forms Builder by using the Run Form button.

4

Creating a Basic Form Module

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Create a form module
- Create a data block
- Save and compile a form module
- Identify Forms file formats and their characteristics
- Describe how to deploy a form module
- Explain how to create documentation for a Forms application

ORACLE

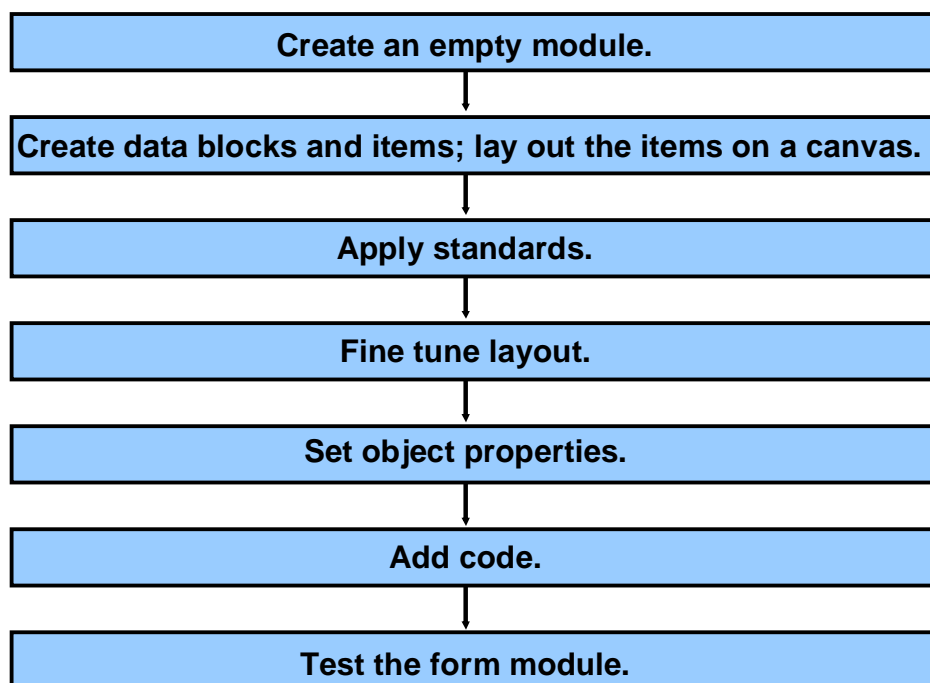
4 - 2

Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

Oracle Forms applications usually consist of one or more form modules. Each module comprises data blocks that are built using table specifications from the database. This lesson shows you how to create a basic form module and its data blocks. You also learn how to deploy a form module.

Designing Form Modules



ORACLE

4 - 3

Copyright © 2009, Oracle. All rights reserved.

Designing Form Modules

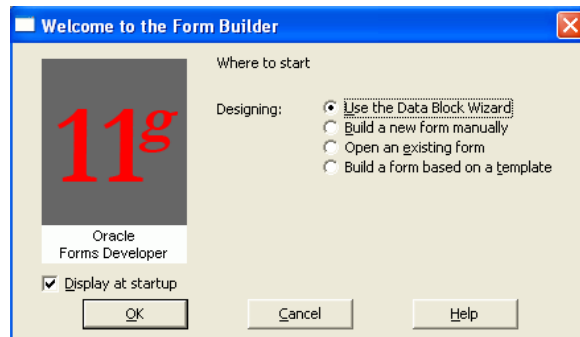
The following actions, along with the tools that are typically used to perform them, are part of creating a form module (although all actions may not be performed for every module):

Action	Forms Builder Tool
Create an empty module.	Object Navigator
Create data blocks and items and arrange the items on a canvas.	Data Block Wizard and Layout Wizard
Apply user interface standards to objects.	Object Library
Fine tune the layout.	Layout Wizard or Layout Editor
Set object properties.	Property Palette
Add code.	PL/SQL Editor
Test the form module.	Run Form button

Creating a New Form Module

Choose one of the following methods:

- Use wizards:
 - The Data Block Wizard
 - The Layout Wizard
- Build the module manually.
- Use the template form.

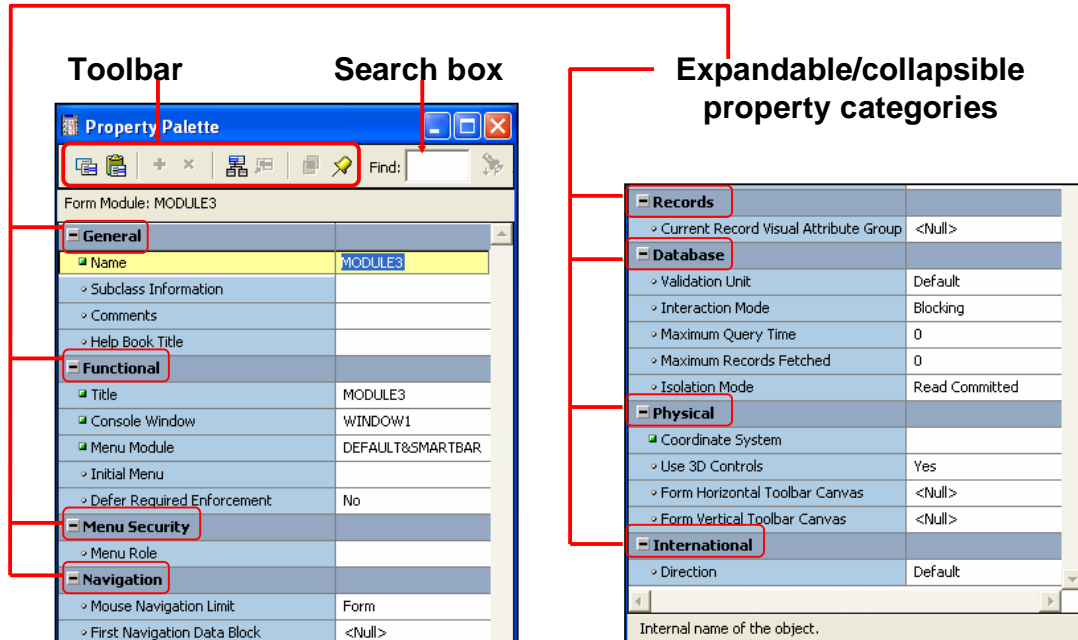


Creating a New Form Module

There are several ways to create a new form module.

- Invoke the Forms Builder component. This takes you to the Forms Builder Welcome page that is shown in the slide (unless you have changed the Preferences to not display it.) Now perform one of the following steps:
 - Select the “Use the Data Block Wizard” option, and follow the required data block creation steps. Then follow the Layout Wizard steps.
 - Select the Build a new form manually option. This takes you to the Forms Builder Object Navigator (automatically creating an empty form module).
 - Select the Build a form based on a template option and use a template form.
- If you are already in the Forms Builder component, you can create a new form module by performing one of the following steps:
 - Double-click the Forms node in the Object Navigator (only when no other form modules are available).
 - Select File > New > Form.
 - Select the Object Navigator node for Forms, and then click the Create icon.

Setting Form Module Properties



Setting Form Module Properties

Each form module consists of several objects. Objects within a form, and the module itself, have properties that define their behavior. You can see the properties of an object and their values in the Property Palette of the object. The screenshots show all the properties of a form module as displayed in the Property Palette.

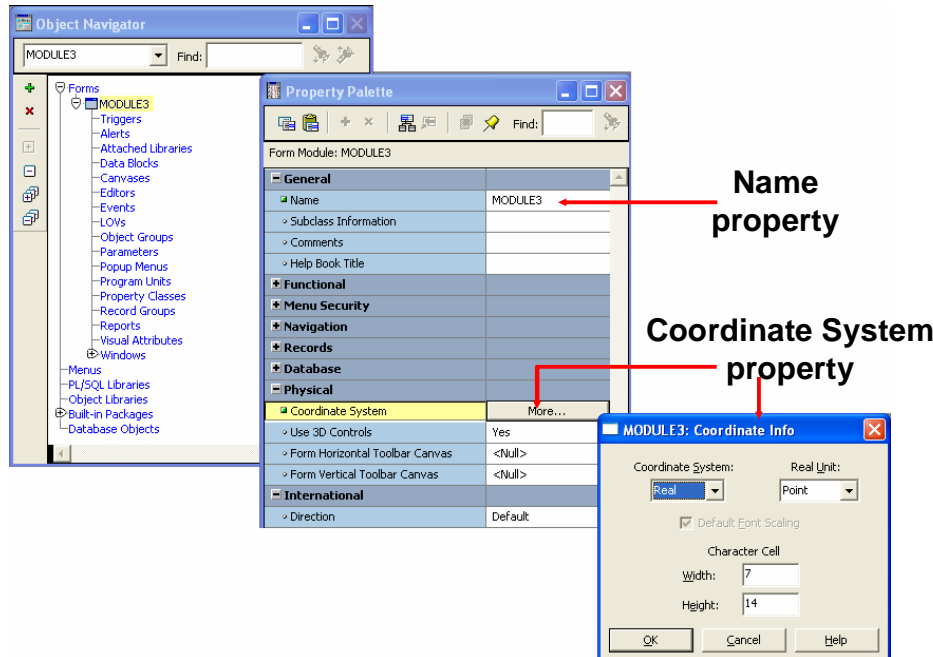
To open the Property Palette of an object, perform one of the following steps:

- Double-click the object's icon in the Object Navigator.
- Double-click the object in the Layout Editor
- Select the object in the Object Navigator and select Tools > Property Palette.
- Right-click the object in the Object Navigator or in the Layout Editor and select Property Palette from the context menu.

A brief description of a selected property appears at the bottom of the palette. To obtain more detailed online Help for any of the properties, select the property and use the Help key (F1), to display a description of that property.

The Property Palette features a toolbar at the top, along with a search box enabling you to easily locate any property. The properties are divided into categories that you can expand or collapse.

Changing the Form Module Name and Coordinate System



Changing the Form Module Name and Coordinate System

You can define the properties of the form module when you first create it or modify the properties later. The properties affect the general behavior of the form and the objects within it. Properties for a form module include the following:

Property	Description
Name	Specifies the internal name of the form module, as it appears in the Object Navigator
Coordinate System	Defines the units used to measure objects in the form and their positions

The screenshots show the module name selected in the Object Navigator and the Property Palette synchronized to display the properties of that selected module. The Property Palette shows the Name property, where you can change the name of the form module. It also shows the Coordinate System property with a More button displayed in the value space. Clicking the button invokes the Coordinate Info dialog box, where you can change the Coordinate System properties.

Changing the Form Module Name and Coordinate System (continued)

Setting the Form Module Name

Forms Builder assigns the name MODULEX to a new form module, where X is the next number available for module names. This name is displayed in the Object Navigator and in the Property Palette.

You should change the default name to a meaningful name in either of the following places:

- In the Object Navigator:
 - Click the form module name.
 - Change the default name as desired and press Enter.
- In the Property Palette (shown in the screenshot)

Note: Follow Oracle naming rules. Do not give two objects of the same type the same name. The name cannot include Oracle or Forms Builder–reserved words.

Choosing a Unit for the Coordinate System

When you click More in the Property Palette window with the Coordinate System property selected, the Coordinate Info dialog box appears.

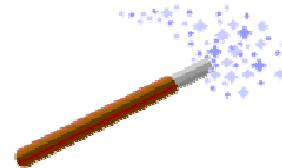
The Coordinate System unit for a form can be one of the following:

- **Real:**
 - Unit can be pixel, centimeter, inch, point, or decipoint.
 - Real units are suitable for GUI applications and enable flexibility and fine alignment when adjusting object positions and sizes.
- **Character:** Units are character cells (default size taken from the default font settings).

The default unit is point (Real). This means that object positions and sizes within the form are measured by this unit. Points provide fine alignment and consistency across different platforms and video devices.

Creating a New Data Block

- Using Forms Builder wizards:
 - Data Block Wizard: Create a data block with associated data source quickly and easily.
 - Layout Wizard: Lay out data block contents for visual presentation.
- Manually
- May require database connection



ORACLE

4 - 8

Copyright © 2009, Oracle. All rights reserved.

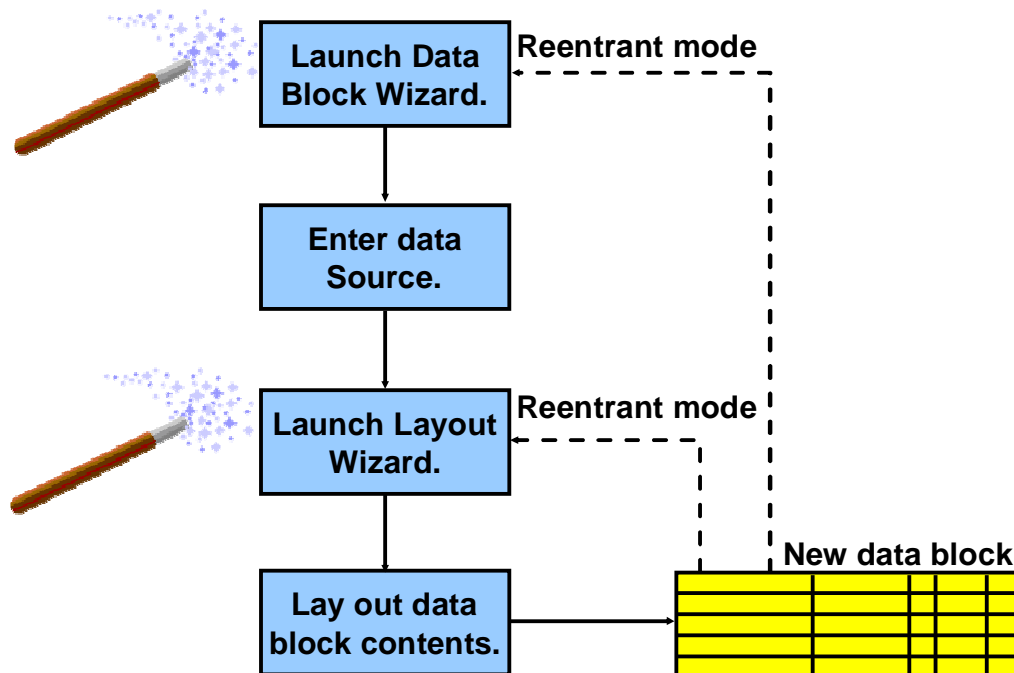
Creating a New Data Block

A form module consists of one or more data blocks and control blocks. Now that you know how to create a new form module, you need to create new data blocks within it.

Block creation involves creating the data block and then laying out its contents for visual presentation. You can create a data block manually or by using the Forms Builder wizards. In this lesson, you learn how to create a new data block based on a database table, using the Data Block Wizard and the Layout Wizard. The slide shows a magic wand, symbolizing a wizard.

Note: Recall that a data block can be based on a table or view, a stored procedure, a FROM clause query, or a transactional trigger. In this course, you use database tables as the source. This requires that you first connect to the database, which you are prompted to do if you use the Data Block Wizard without first connecting. Connecting to the database was covered in the previous lesson.

Using the Wizards to Create a Data Block



4 - 9

Copyright © 2009, Oracle. All rights reserved.

ORACLE

Using the Wizards to Create a Data Block

The slide shows the process of creating a new data block by launching the Data Block Wizard, entering the data source, launching the Layout Wizard, and laying out the contents of the data block. You can invoke either wizard in reentrant mode.

Data Block Wizard

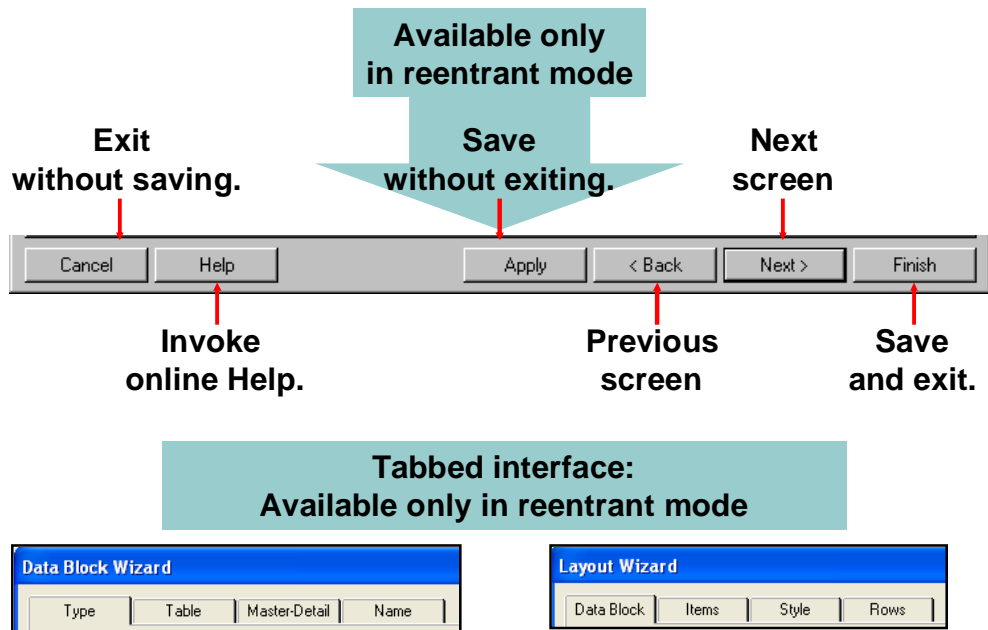
The Data Block Wizard enables you to create (or modify) data blocks quickly and easily for use in your application. The wizard can automatically generate code to enforce integrity constraints in the database.

Layout Wizard

Although the Data Block Wizard enables you to create a new data block easily with its associated data sources, it does not deal with the visual presentation of objects included in the data block. After you create the data block, you need to lay out its contents for user interaction. To accomplish this task quickly and easily, use the Layout Wizard.

Note: The wizards are not the only way to perform a task such as building a data block, but they are usually the simplest. You can build a block manually instead of using the wizards.

Navigating the Wizards



ORACLE

4 - 10

Copyright © 2009, Oracle. All rights reserved.

Navigating the Wizards

The Data Block Wizard and the Layout Wizard provide buttons as shown in the slide:

Button	Description
Cancel	Cancels any changes and exits the wizard
Help	Displays online Help text for the current wizard page
Back	Navigates to the previous page in the wizard
Next	Navigates to the next page in the wizard
Apply	Applies your changes without exiting the wizard (available only in reentrant mode)
Finish	Saves any changes and exits the wizard

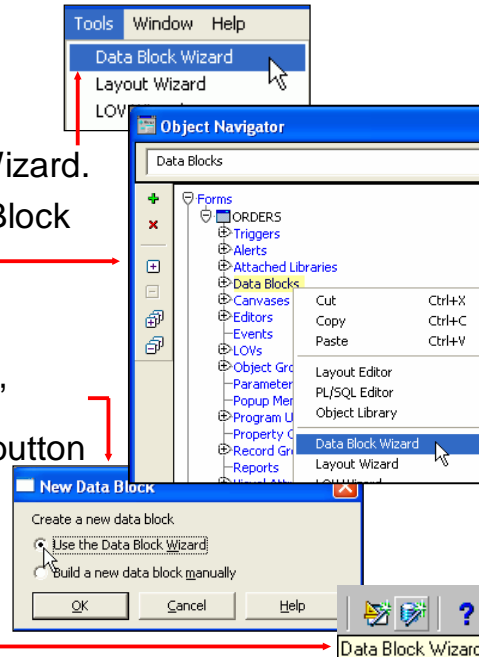
If you click Next or Back before entering all necessary information for a particular wizard page, the wizard prevents you from navigating to another page. Similarly, if you have not entered all necessary information when you click Apply or Finish, the wizard automatically takes you to the page where you can finish entering the required information.

In reentrant mode, which enables you to modify existing blocks or layouts, the wizards have a tabbed interface that enables you to quickly navigate to the section you want to modify. You learn more about modifying data blocks and layouts in the lesson titled “Creating a Master-Detail Form.”

Launching the Data Block Wizard

In Forms Builder, do one of the following:

- Select Tools > Data Block Wizard.
- Right-click and select Data Block Wizard.
- Select the Data Blocks node and click Create icon; select “Use the Data Block Wizard.”
- Use the Data Block Wizard button on the toolbar.



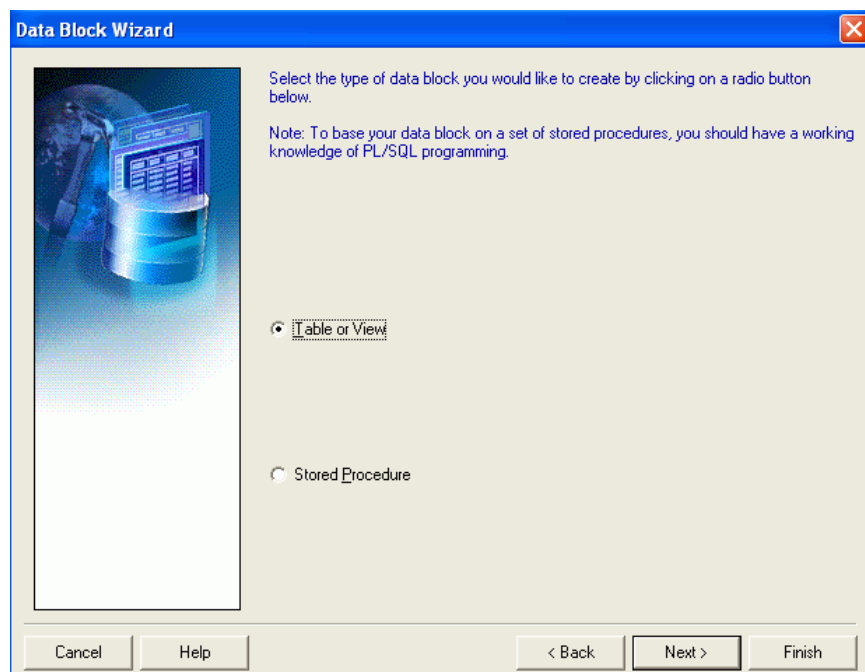
Launching the Data Block Wizard

To launch the Data Block Wizard to create a new data block, perform one of the following:

- In the Forms Builder, do one of the following:
 - Select Tools > Data Block Wizard from the Forms Builder default menu system.
 - Right-click and select Data Block Wizard.
 - In the Object Navigator, select the Data Blocks node, then click the Create icon. In the New Data Block dialog box, select the “Use the Data Block Wizard” option.
 - Click Data Block Wizard on the toolbar.
- If you are not already in Forms Builder, launch Forms Builder and select the “Use the Data Block Wizard” option on the Forms Builder Welcome page.

The screenshots show the Tools menu, the context menu, the Data Block Wizard option, and the toolbar.

Data Block Wizard: Type Page



ORACLE

4 - 12

Copyright © 2009, Oracle. All rights reserved.

Data Block Wizard: Type Page

Use the Data Block Wizard to create a new data block with its associated data sources. The Data Block Wizard consists of several pages. To create a new data block, you must interact with each page.

Welcome Page

Click Next to continue.

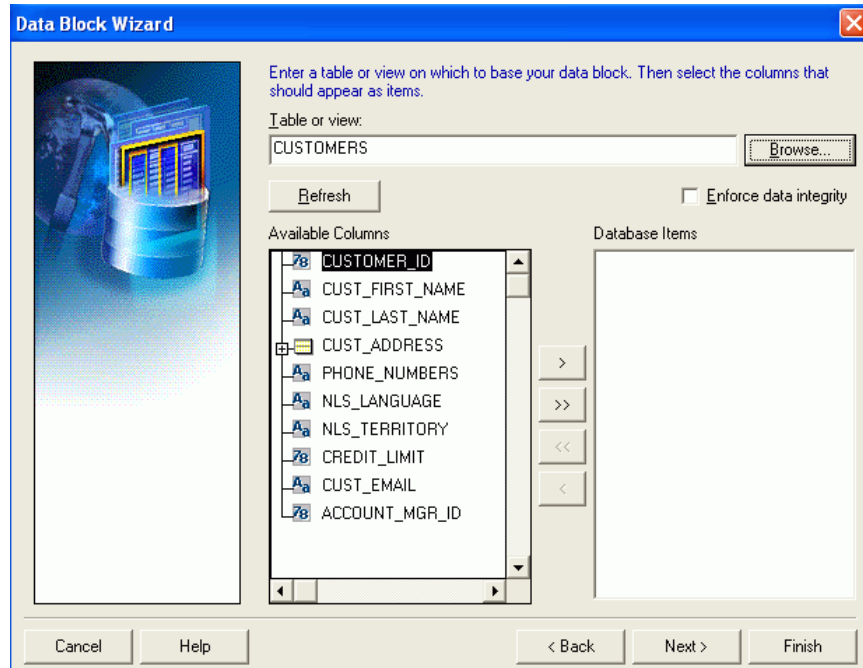
Type Page (shown in the screenshot)

Choose between one of two data source types:

- Table or View
- Stored Procedure

Select the Table or View (default) option.

Data Block Wizard: Table Page



Data Block Wizard: Table Page

On the Table page (shown in the screenshot):

1. Enter the table or view name for the data source name, or click Browse and select a name from a dialog box.
2. Click Refresh to display a list of columns in the selected table or view. If you are not connected to the database, the Connect box is displayed.
3. Select the columns you want to include in the data block. (To select more than one column, press and hold Ctrl and then select the columns.)
4. Click the double-right arrow or the double-left arrow to include or exclude all columns, or click the right arrow or the left arrow to include or exclude selected columns only.
5. Select the Enforce data integrity check box if you want the wizard to enforce the database integrity constraints.

Note: If there is at least one other existing block in the current module, the next page that displays is the Master-Detail page, where you can associate the new data block with other master data blocks. This page is discussed in the lesson titled “Creating a Master-Detail Form.”

Data Block Wizard: Finish Page



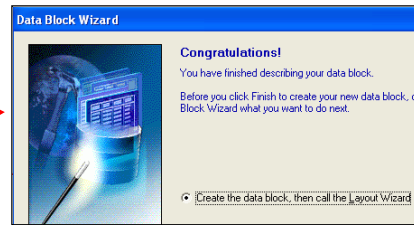
Data Block Wizard: Finish Page

On the Finish Page (shown in the screenshot) select the “Create the data block, then call the Layout Wizard” option. Click Finish to create the new data block and immediately invoke the Layout Wizard.

Note: You have the option of exiting the Data Block Wizard at this stage, without immediately invoking the Layout Wizard. If you do so, you can either lay out the data block manually or invoke the Layout Wizard at a later time to lay out the items of a data block. To invoke the Layout Wizard at a later time, select the data block in the Object Navigator, and then select Tools > Layout Wizard.

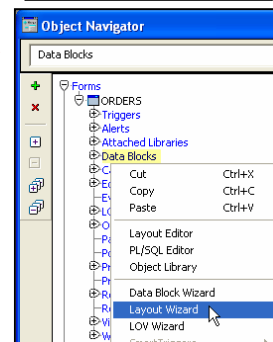
Launching the Layout Wizard

- Launch automatically from the Data Block Wizard.



- In Forms Builder, do one of the following:

- Select Tools > Layout Wizard.
- Right-click and select Layout Wizard.
- Use the Layout Wizard button on the toolbar.



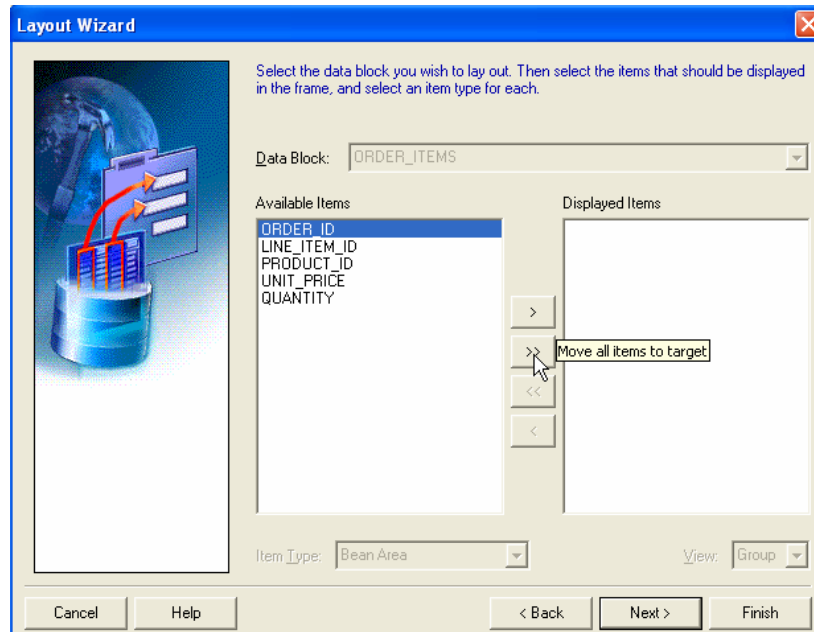
Launching the Layout Wizard

To launch the Layout Wizard to create a new layout, perform one of the following steps:

- Launch automatically from Data Block Wizard.
- In the Forms Builder, do one of the following:
 - Select Tools > Layout Wizard from the Forms Builder default menu system.
 - Right-click and select the Layout Wizard option.
 - Click Layout Wizard on the toolbar.

The screenshot at the top of the slide shows the last page of the Data Block Wizard that can optionally launch the Layout Wizard. The other screenshots show invoking the Layout Wizard from Tools menu, the context menu, and the toolbar.

Using the Layout Wizard: Canvas and Data Block Pages



Using the Layout Wizard: Canvas and Data Block Pages

Use the Layout Wizard to lay out the data block items for visual presentation quickly and easily. The Layout Wizard consists of several pages. You must interact with each page.

Canvas Page

1. Select New Canvas from the Canvas pop-up list to get a new canvas on which to display the data block items.
2. Select Content as the canvas type in the Type pop-up list.

Data Block Page (shown in the screenshot)

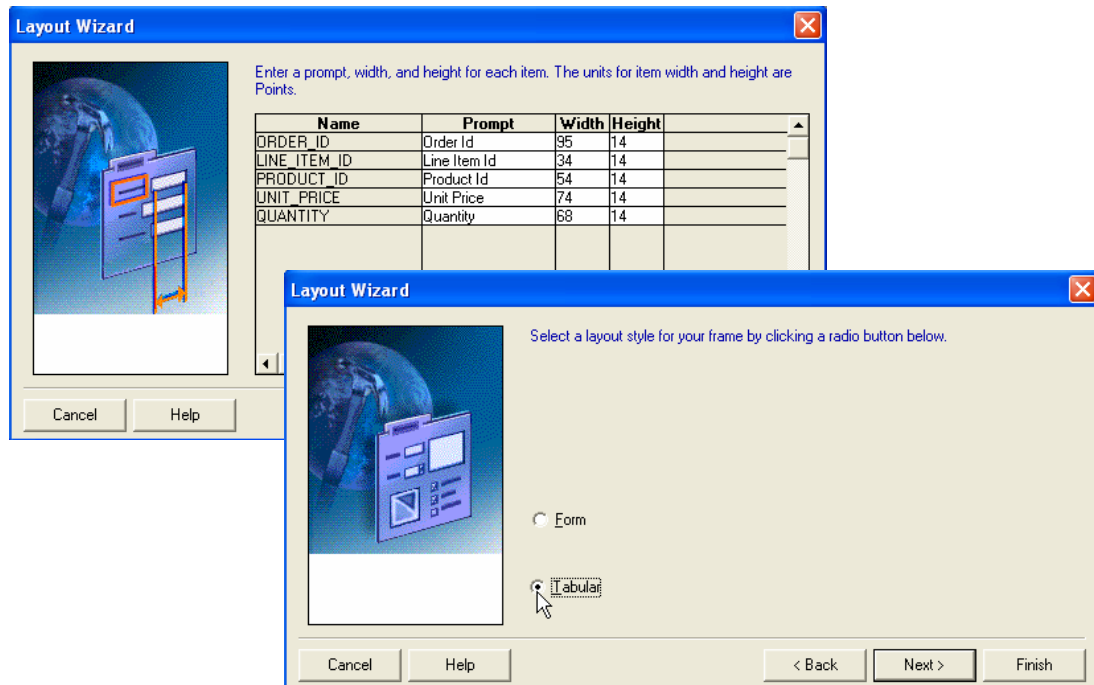
1. Select the items that you want to display in the data block frame. (To select more than one column, press and hold Ctrl and then select the columns.)
2. Click the double-right arrow or double-left arrow to include or exclude all items, or click the right arrow or the left arrow to include or exclude selected items only. You can also drag selected items from one list to another.

Note: To lay out the items in a particular sequence, drag the items into that sequence.

3. You can use the Item Type pop-up list to select a type for each item. The default type is Text for each item.

Note: An item type can also be changed later to something else, such as a pop-up list or a radio group.

Using the Layout Wizard: Items and Style Pages



ORACLE

4 - 17

Copyright © 2009, Oracle. All rights reserved.

Using the Layout Wizard: Items and Style Pages

Items Page

Specify prompt text and display width and height for each display item.

Style Page

Select a layout style for your frame. Your options are:

- Form (usually used to create single-record data blocks)
- Tabular (usually used to create multirecord data blocks)

The slide shows both the Items and the Style pages.

Using the Layout Wizard: Records and Finish Pages

Layout Wizard

Enter a title for the frame. Also be sure to specify the number of database records to be displayed in the frame, as well as the distance between each record.

To display a scrollbar in the frame that can be used to scroll through database records, check the 'Display Scrollbar' check box.

Frame Title:

Records Displayed:

Distance Between Records:

☐ Display Scrollbar

Cancel Help < Back Next > Finish

ORACLE

4 - 18

Copyright © 2009, Oracle. All rights reserved.

Using the Layout Wizard: Records and Finish Pages

Records Page (shown in the screenshot)

1. Enter a title in the Frame Title field.
2. Enter the number of records that you want to display at run time in the Records Displayed field.
3. Enter the physical distance (in the coordinate system unit of the form) between records if you are displaying more than one record at a time.
4. You can select the Display Scrollbar check box to display a scroll bar next to the frame (common for multirecord data blocks).

Finish Page

Click Finish to create a new frame and lay out the selected items for the new data block. The Layout Wizard steps are complete.

Note: After you complete the Layout Wizard steps, you can view the layout in the Layout Editor, where you can customize or modify the layout if necessary.

Data Block Functionality

After you create a data block with the wizards, Forms Builder automatically creates:

- A form module with database functionality including query, insert, update, and delete
- A frame object
- Items in the data block
- A prompt for each item
- Triggers needed to enforce database constraints if the Enforce data integrity check box is selected

ORACLE

4 - 19

Copyright © 2009, Oracle. All rights reserved.

Data Block Functionality

After you create a new data block by using the wizards, Forms Builder automatically creates the following objects for you:

- A new form module with a default menu (Basic database functionality such as querying, inserting, updating, and deleting is automatically available on the items in the base-table block when you run the new form.)
- The new data block is created with default property values. These values can be modified to change the behavior of the form.
- A frame object to arrange the items within the new data block
- An item for each database table column included in the data block (Each item is assigned default property values to match the underlying column specifications.)
- A prompt for each item in the data block (The default prompt is the name of the column.)

In addition, Forms Builder may create triggers to validate user input if you select the Enforce data integrity check box on the Table page of the Data Block Wizard.

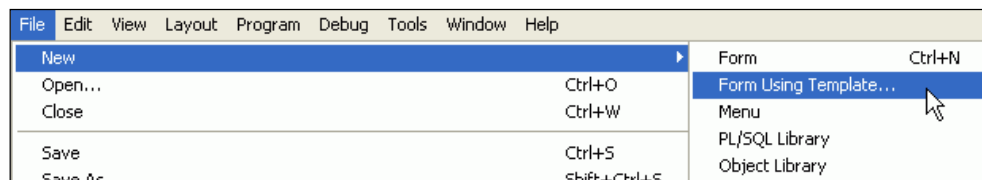
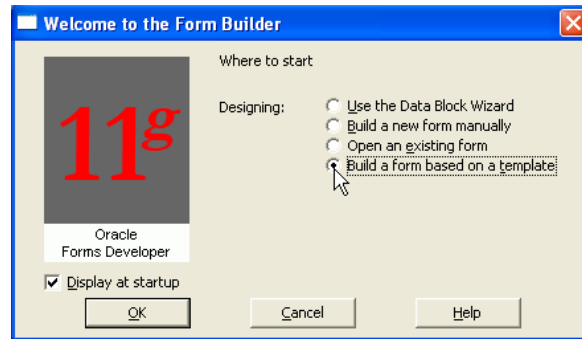
Template Forms

To use a form template:

- Select the template option from the Welcome dialog box

Or

- Select File > New > Form Using Template



ORACLE

4 - 20

Copyright © 2009, Oracle. All rights reserved.

Template Forms

You can create a new form based on standard template forms, so that you can provide other team members with a default starting point. Templates typically include generic objects, such as graphics, toolbars, and program units. You can define standard window layouts, standard toolbars, and other common objects that you want to include in new forms.

Creating a Form Based on a Template

To create a form based on a template, perform one of the following:

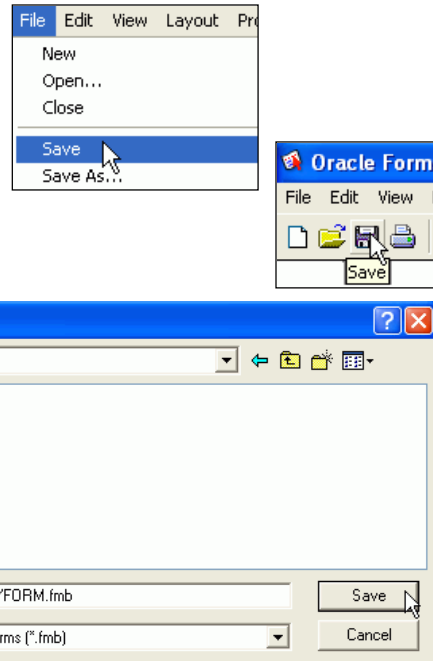
- Start Forms Builder. In the “Welcome to the Form Builder” dialog box, select the Build a form based on a template option, and then click OK.
- Select File > New > Form Using Template from the menu.

After choosing to use a template, Forms Builder presents you with a file dialog where you can browse to the form you want to use as a template.

Saving a Form Module

To save the form module:

- Select File > Save
or
Click the Save icon
- Enter a file name
- Navigate to the desired location
- Click Save



ORACLE

4 - 21

Copyright © 2009, Oracle. All rights reserved.

Saving a Form Module

To save the form module definition, perform one of the following steps:

- Select File > Save.
- Click the Save icon on the toolbar.

Both of these options display the Save As dialog box for the initial save. In the dialog box, do the following:

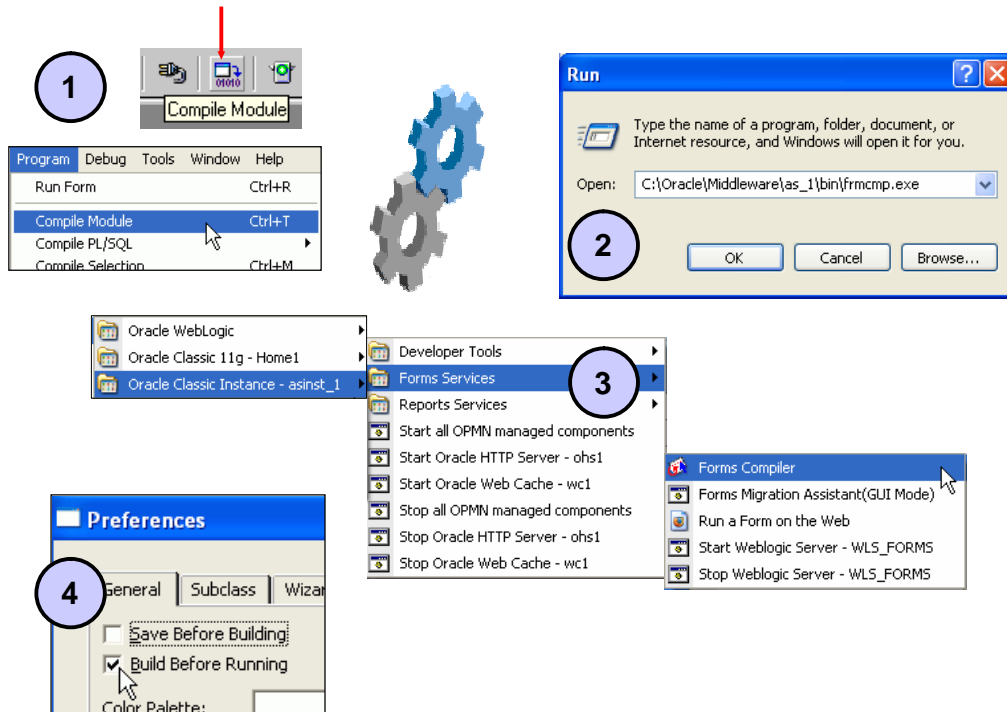
1. Enter a file name.
2. Navigate to the directory where you want to save the file.
3. Click Save.

The screenshots show the menu option, the toolbar, and the Save As dialog box.

Note

- When you save a form, a .fmb file is produced. This saved definition of a form in the file system is not executable, and can be opened only by Forms Builder.
- When you work with more than one module at a time, Forms Builder separately keeps track of the changes that you make to each module. When you execute a Save command, only the current module is saved.

Compiling a Form Module



ORACLE

4 - 22

Copyright © 2009, Oracle. All rights reserved.

Compiling a Form Module







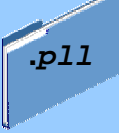


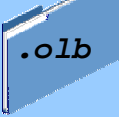

Before you can run a form, you must compile an executable (.fmx) file from the design (.fmb) file that you created in the Forms Builder. Compiling a form (or menu) module creates the needed executable file. There are several ways to compile a form:

	Action	Type of Compilation
1.	With module open in Forms Builder, select Program > Compile Module (or click the Compile Module icon).	Explicit
2.	Launch the Forms Compiler component from the command line or the Windows Run dialog box (frmcmp.exe).	Explicit
3.	Launch the Forms Compiler component from the Windows Start menu.	Explicit
4.	Set the Build Before Running preference.	Implicit

Each of these options is depicted in the screenshots in the slide.

Note: Compiling and saving are two independent tasks. Performing one does not automatically accomplish the other.

Module Types and Storage Formats

Form module			
Menu module			
PL/SQL Library			
Object Library			

ORACLE

4 - 23

Copyright © 2009, Oracle. All rights reserved.

Module Types and Storage Formats

When you create form modules, menu modules, and library documents in the Forms Builder, they are stored in source files (`.fmb`, `.mmb`, and `.pll`) that have a binary format and are portable across platforms. The executable application files (`.fmx`, `.mmx`, and `.plx`) are also in a binary format; however, they are not portable across platforms.

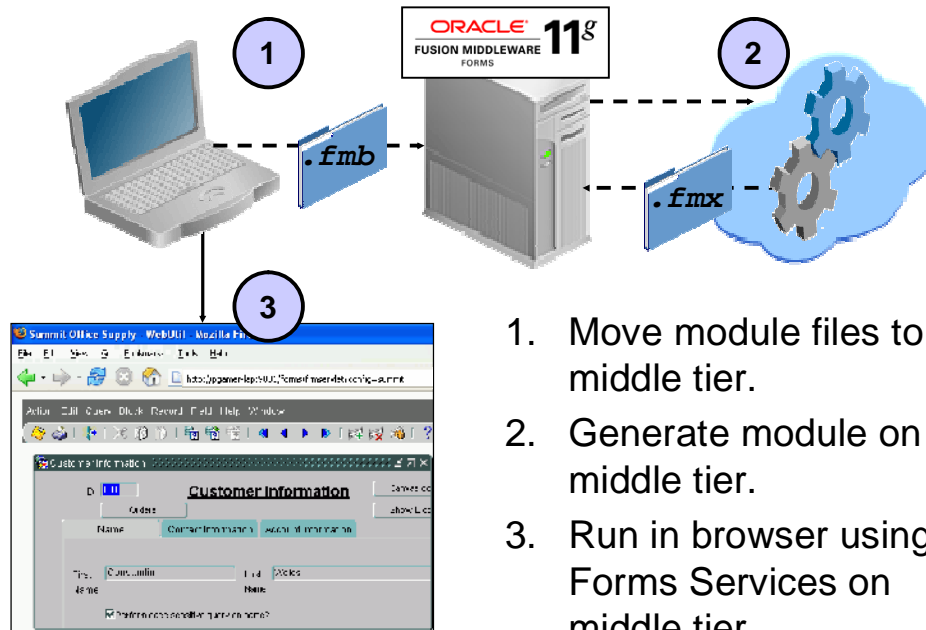
The graphics in the slide show the various file types. The first column of graphics shows the portable binary files, the second column of graphics shows the executables, and the third column of graphics shows the text files that you can create for form modules, menus, PL/SQL libraries, and object libraries.

Module Types and Storage Formats (continued)

Module/ Document	Extension	Storage Format	Portable
Form	.fmb	Form module binary	Yes
	.fmx	Form module executable; executable	No
	.fmt	Form module text	Yes
Menu	.mmb	Menu module binary	Yes
	.mmx	Menu module executable; executable	No
	.mmt	Menu module text	Yes
PL/SQL Library	.pll	PL/SQL Library document binary	Yes
	.plx	PL/SQL Library document executable (no source)	No
	.pld	PL/SQL Library document text	Yes
Object Library	.olb	Object Library module binary	Yes
	.olt	Object Library module text	Yes

Note: .pll is portable but requires recompilation because it contains both source and compiled code.

Deploying a Form Module



1. Move module files to middle tier.
2. Generate module on middle tier.
3. Run in browser using Forms Services on middle tier.

ORACLE

Deploying a Form Module

Although you may test a form on your client machine, for production applications you usually deploy the module to a middle-tier machine. This machine may be running on a different platform; if so, you need to recompile the module after you transfer it to the middle tier.

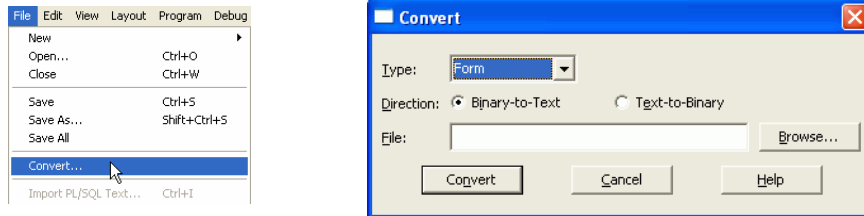
You can use an FTP utility to move the `.fmb` and other needed files to the middle-tier machine, into a directory specified in `FORMS_PATH`, as shown in the slide. If the platform is the same as your development platform, you can move the `.fmx` and other executable files there as well. If it is a different platform, you can invoke the Forms Compiler on the middle tier to recompile the module files, as shown in the slide.

After the executables have been placed on the middle tier, you can invoke the application in a browser, using a URL that points to the Forms Servlet on the middle-tier Web server.

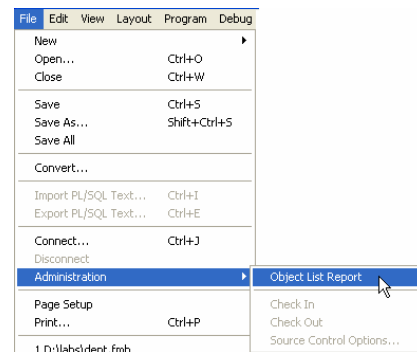
Note: `FORMS_PATH` and other environment variables are set for run time on the middle tier as described in the lesson titled “Working in the Forms Environment.”

Producing Text Files and Documentation

- Convert a binary file to a text file.



- Create an ASCII file for a form module.



Producing Text Files and Documentation

The files normally produced by saving and generating modules are in binary format. To convert a binary file to text, perform the following:

1. Select File > Convert, as shown in the first screenshot in the slide.
This opens the Convert dialog box.
2. In the Convert dialog box:
 - a) Select the type of module (Form, Menu, PL/SQL Libraries, and Object Libraries), the file to convert, and the direction (Binary-to-Text).
 - b) Click Convert. This produces a text file for the module with the name <module>.fmt.

To produce documentation for your module, perform the following:

1. Select the module to be documented in the Object Navigator.
2. Select File > Administration > Object List Report. This produces an ASCII file with the name <module>.txt.

You can also produce documentation in other ways not covered in this course:

- Use Forms API to produce custom documentation of the module.
- Convert the module to a .xml file with a separate utility included with Forms.

Summary

In this lesson, you should have learned that:

- To create a form module, you create an empty module, then add data blocks and other elements
- You can create a data block manually or with the Data Block Wizard and the Layout Wizard
- You can save and compile a form module using the File and Program menus or from the toolbar
- You can store form, menu, and library modules in text format (useful for documentation), in a portable binary format, or a nonportable binary executable format
- To deploy a form module, you move it to the application server machine and also may need to generate it

ORACLE

4 - 27

Copyright © 2009, Oracle. All rights reserved.

Summary

In this lesson, you should have learned about:

- Building a new form module by using the following methods:
 - Forms Builder wizards
 - Manually
 - Template form
- Using the Data Block Wizard to create a new data block with its associated data sources quickly and easily
- Using the Layout Wizard to quickly lay out the new data block contents for user interaction
- Saving the form module to preserve its definition; compiling it to get an executable file; running the form module to test it
- Using several module types and storage formats that are available for form modules, menu modules, PL/SQL Library documents, and Object Library modules, including a text format that can be used for documentation

Practice 4: Overview

This practice covers the following topics:

- Creating a new form module
- Creating a data block by using Forms Builder wizards
- Saving and running the form module

ORACLE

4 - 28

Copyright © 2009, Oracle. All rights reserved.

Practice 4: Overview

In this practice, you create a new form module. You create a single-block form that displays a single record.

5

Creating a Master-Detail Form

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Create data blocks with relationships
- Modify a data block
- Modify the layout of a data block
- Run a master-detail form

ORACLE

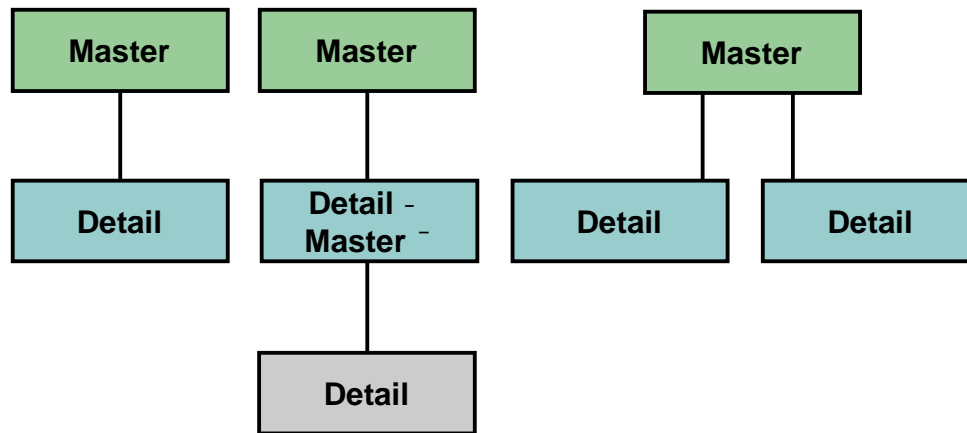
5 - 2

Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

A common type of application is to show a record or records from one table along with associated records from another table. Forms Developer gives you the ability to quickly define additional data blocks and to define relations between the new blocks and any existing blocks in the form. This lesson shows you how to create a master-detail form and how to modify a data block and its layout.

Creating Data Blocks with Relationships



ORACLE

5 - 3

Copyright © 2009, Oracle. All rights reserved.

Creating Data Blocks with Relationships

A form module can contain one or more data blocks. Each data block can stand alone or be related to another data block. The slide shows the following:

- A master block with one detail block
- A master block with one detail block that is the master for another detail block
- A master block with two detail blocks

Master-Detail Relationship

A master-detail relationship is an association between two data blocks that reflect a primary key/foreign key relationship between the database tables on which the two data blocks are based. The master data block is based on the table with the primary key, and the detail data block is based on the table with the foreign key. A master-detail relationship equates to the one-to-many relationship in the entity relationship diagram.

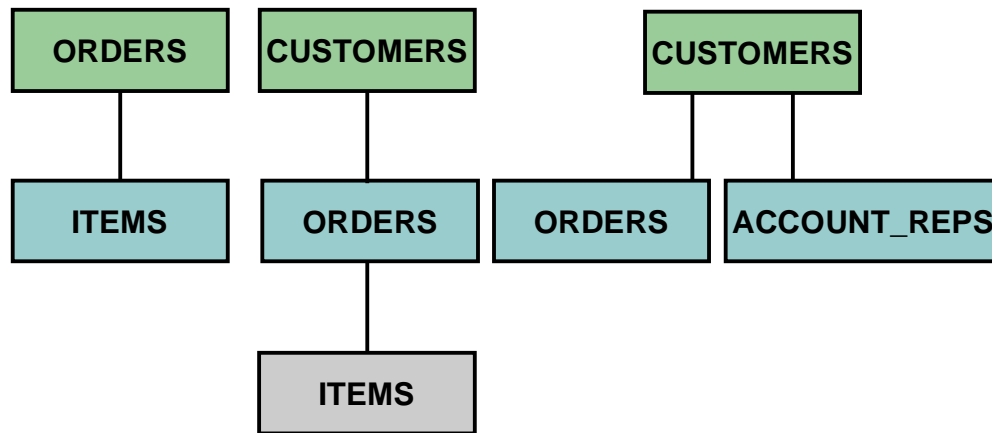
A Detail Block Can Be a Master

You can create block relationships in which the detail of one master-detail link is the master for another link.

A Master Block Can Have More Details

You can create more than one detail blocks for a master block.

Block Relationship: Examples



ORACLE

5 - 4

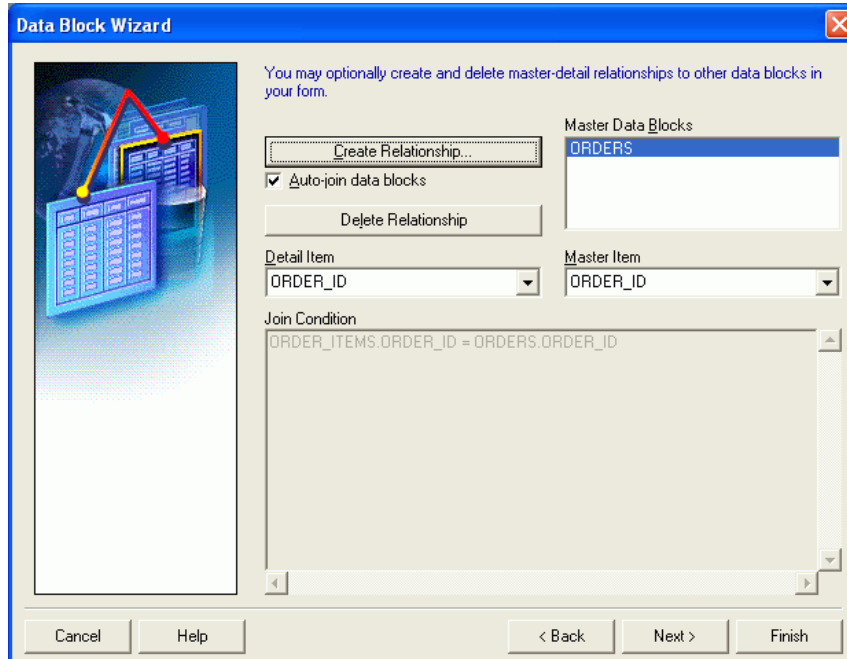
Copyright © 2009, Oracle. All rights reserved.

Blocks Relationship: Examples

The following are examples of the master-detail structure:

- **Master-detail:** ORDERS to ITEMS, shown at the left of the slide
- **Master-detail-detail:** CUSTOMERS to ORDERS to ITEMS, shown in the middle of the slide
- **Master-2*detail:** CUSTOMERS to ORDERS and CUSTOMERS to ACCOUNT_REPS, shown at the right of the slide

Data Block Wizard: Master-Detail Page



Data Block Wizard: Master-Detail Page

You can build a master-detail form module either by creating a relation between a master and detail block explicitly or by using the Data Block Wizard to create it implicitly.

1. Create the master block as described in the lesson titled “Creating a Basic Form Module.”
2. Invoke the Data Block Wizard in the Object Navigator.
3. Follow the same steps as before to create a new data block in the Data Block Wizard until you come to the Master-Detail page, shown in the slide. On this page, select the Auto-join data blocks check box and click Create Relationship.

Note: If the Auto-join data blocks check box is not selected, the Data Block dialog box is displayed with a list of all data blocks in the form without any foreign key constraint names.

Data Block Wizard: Master-Detail Page (continued)

4. Select a master data block in the Data Block dialog box and click OK. The wizard automatically creates the join condition between the detail and master data blocks in the Join Condition field and displays the name of the master data block in the Master Data Blocks field.

Note: If the Auto-join data blocks check box is not selected, the wizard does not automatically create the join condition between the detail and master data blocks. You must use the Detail Item and Master Item pop-up lists to create a join condition manually.

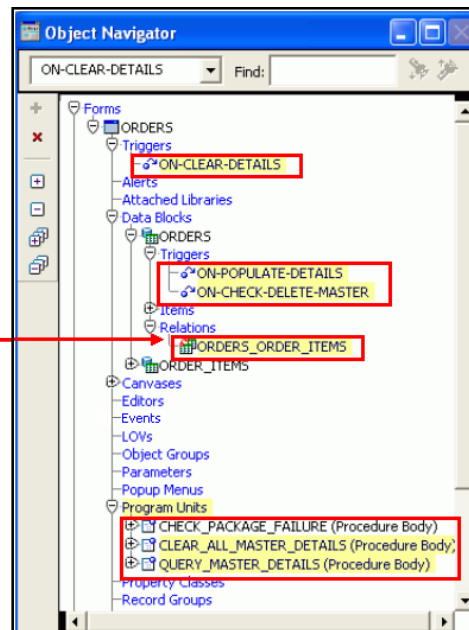
5. Click Next, and then complete the Data Block Wizard steps. Perform the Layout Wizard steps as described earlier in the lesson titled “Creating a Basic Form Module” to finish creating and laying out the detail data block.

Note: The master data block must exist in the form module before you create the detail block.

You can also create a relation by invoking the Data Block Wizard in reentrant mode.

Examining the New Relation

- New relation object created in Object Navigator under master data block node
- Default name assigned: *<MasterDataBlock>_<DetailDataBlock>*
- Triggers and program units generated automatically



ORACLE

Examining the New Relation

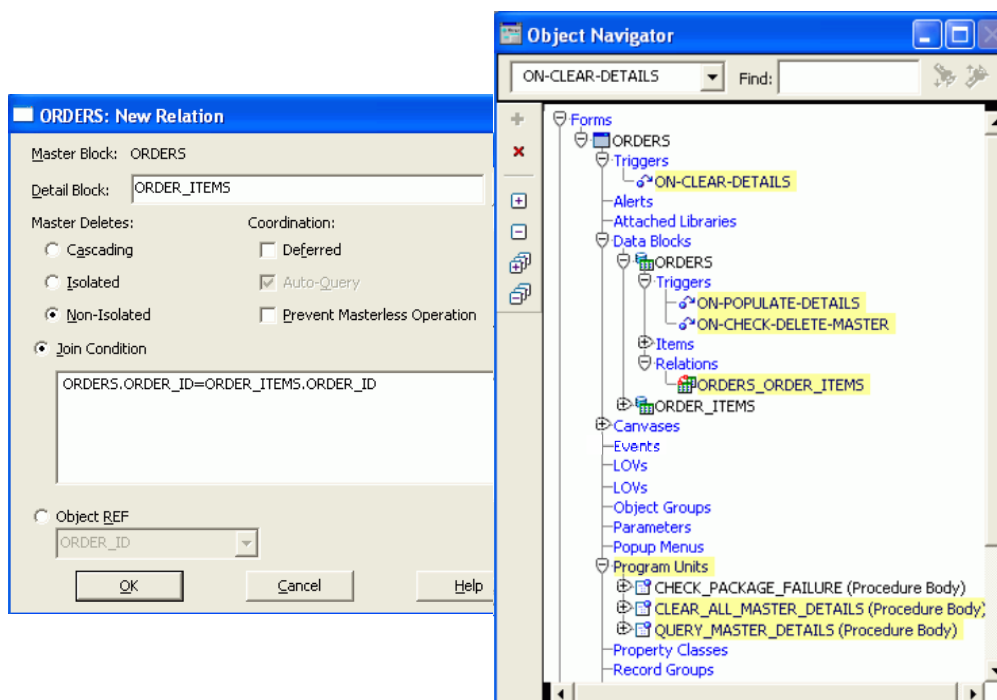
After you create a master-detail form module, the Data Block Wizard automatically creates a form object that handles the relationship between two associated data blocks. This object is called a *relation*. The following tasks occur automatically:

- The new relation object is created under the master data block node in the Object Navigator with default properties.
- The relation is given the following default name: *<MasterDataBlock>_<DetailDataBlock>*—for example, CUSTOMERS_ORDERS.
- Triggers and program units are generated to maintain coordination between the two data blocks.

The screenshot shows the following objects that are created in the Object Navigator for a relation:

- A form-level On-Clear-Details trigger (you learn more about triggers in the lesson titled “Introduction to Triggers”)
- Block-level triggers for the master block: On-Populate-Details and On-Check-Delete-Master
- The relation itself
- Three program units: CHECK_PACKAGE_FAILURE, CLEAR_ALL_MASTER_DETAILS, and QUERY_MASTER_DETAILS

Creating a Relation Manually



Creating a Relation Manually

You can create a relation either implicitly with the Data Block Wizard, or explicitly in the Object Navigator.

Explicit Relations

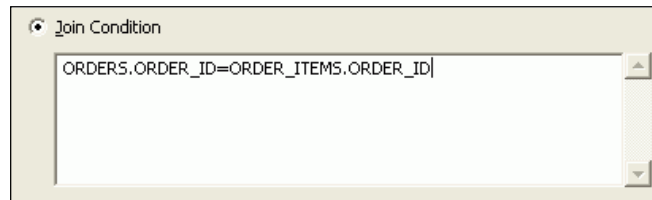
If a relation is not established when default blocks are created, you can create your own. To explicitly create a relation, perform the following steps:

1. Select the Relations node under the master block entry in the Object Navigator.
2. Click the Create icon. The New Relation window is displayed.
3. Specify the name of the detail block.
4. Choose your master delete property.
5. Choose your coordination property.
6. Specify the join condition.
7. Click OK.

The new relation, new triggers, and new program units are highlighted in the Object Navigator. Like implicitly created relations, the relation is given the default name of MasterDataBlock_DetailDataBlock—for example, CUSTOMERS_ORDERS. The same PL/SQL program units and triggers are created automatically when you explicitly create a relation as when the relation is created implicitly.

Defining the Join Condition

- The join condition creates a primary key/foreign key link between blocks.
- Define a join condition by using:
 - Block and item names (not table and column names). Do not precede names with colon.
 - SQL equijoin syntax



ORACLE

5 - 9

Copyright © 2009, Oracle. All rights reserved.

Defining the Join Condition

Use a join condition to:

- Create links between blocks using SQL
- Alter links between blocks using SQL

Define a join condition using:

- Usual SQL equijoin condition syntax (necessary because Forms copies the value from the master block to the related item in the detail block)
- Block names instead of the base table names (do not precede with colon)
- Item names that exist in the form module rather than base table column names

The screenshot shows the part of the New Relation dialog box where you define the join condition. The example shows the condition:

`ORDERS.ORDER_ID=ORDER_ITEMS.ORDER_ID`

Running a Master-Detail Form Module

- Automatic block linking for:

- Querying
- Inserting

- Default deletion rules:
Cannot delete master
record if detail records
exist

The screenshot displays a master-detail form. The master block contains fields for 'Order Id' (2458), 'Customer Id' (101), 'Sales Rep Id' (153), 'Order Date', and 'Order Status'. The detail block, titled 'Order Items', contains a table with columns 'Order Id', 'Line Item Id', 'Product Id', and 'U'. The table lists five items for Order Id 2458.

Order Id	Line Item Id	Product Id	U
2458	1	3117	38
2458	2	3123	79
2458	3	3127	488.4
2458	4	3134	17
2458	5	3143	15

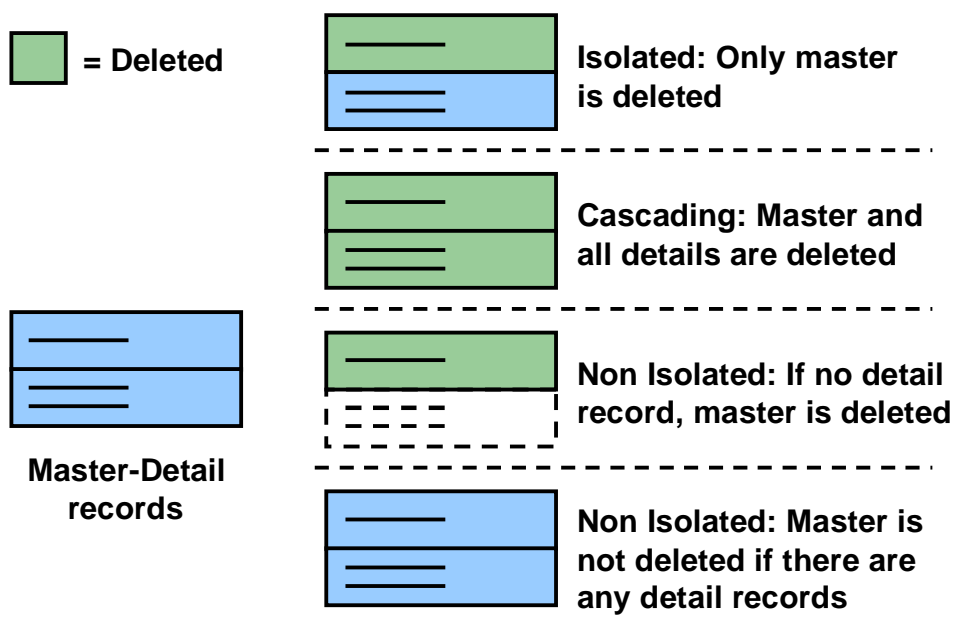
Running a Master-Detail Form Module

When you run your master-detail form module, you will find that:

- Querying the master data block immediately by default retrieves corresponding detail records, as shown in the screenshot
- Deleting a master record is prevented if detail records exist
- Inserting a detail record automatically associates it with the currently displayed master

Note: You can change the above behavior by modifying the relation object properties.

Setting Delete Record Behavior



ORACLE

5 - 11

Copyright © 2009, Oracle. All rights reserved.

Setting Delete Record Behavior

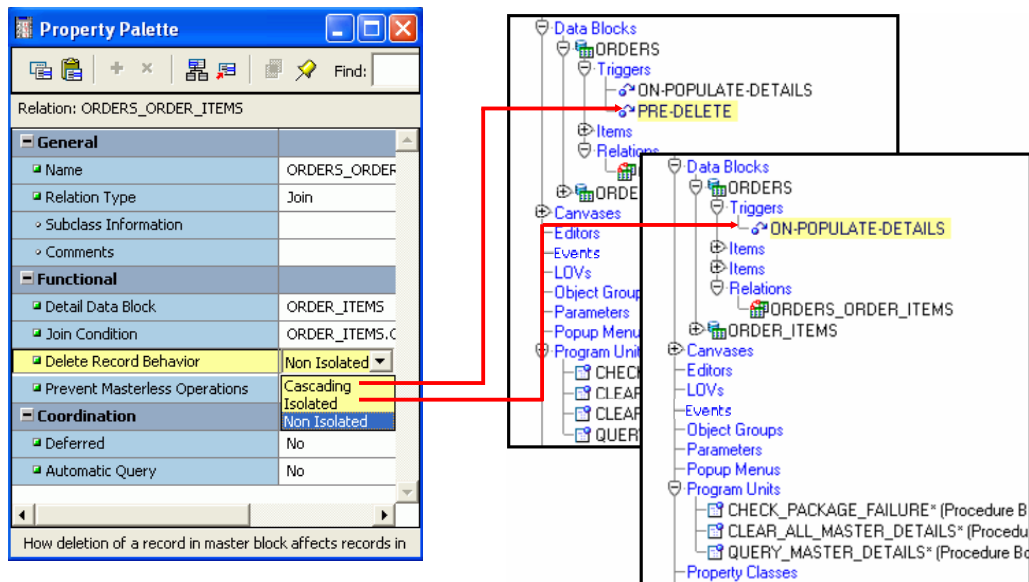
By setting the Delete Record Behavior property, you can prevent, propagate, or isolate deletion of a record in a master block when corresponding records exist in the detail block. For example, you can delete all corresponding line items when an order is deleted.

Property Value	Use
Non Isolated	Prevents the deletion of the master record when detail records exist; the master record is deleted only if no detail records exist
Cascading	Deletes the detail records when a master record is deleted
Isolated	Deletes only the master record

The graphics show what happens to master and detail records with different settings of the Delete Record Behavior property, as described in the text beside each graphic.

Note: Although deleting with the cascading property may remove many detail records, the commit message shows only the number of records deleted from the master block.

Delete Record Behavior and Triggers



ORACLE

5 - 12

Copyright © 2009, Oracle. All rights reserved.

Delete Record Behavior and Triggers

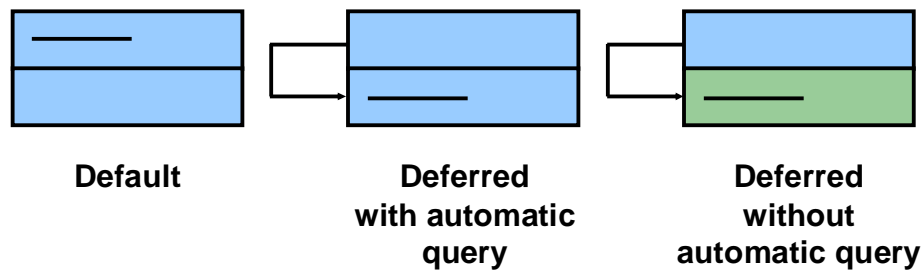
Changing the Delete Record Behavior property value affects the block-level triggers of the master block as follows:

- If you change the Delete Record Behavior property from the default of Non Isolated to Cascading, the On-Check-Delete-Master trigger is replaced with the Pre-Delete trigger.
- If you change the Delete Record Behavior property from the default of Non Isolated to Isolated, the On-Check-Delete-Master trigger is removed.

The screenshots show the block-level triggers in the Object Navigator when the Delete Record Behavior property is set to:

- **Cascading:** On-Populate-Details and Pre-Delete
- **Isolated:** On-Populate-Details only

Setting Coordination Properties



ORACLE

5 - 13

Copyright © 2009, Oracle. All rights reserved.

Setting Coordination Properties

Setting the two coordination properties (Deferred and Automatic Query) controls how the detail records are displayed when a master block is queried. For example, you can defer querying the line items for an order until the operator navigates to the item block.

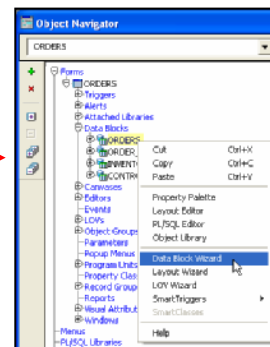
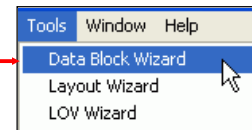
Coordination Property	Use
Default (depicted in first slide graphic)	Forces coordination of blocks to occur whenever the master record is changed by a user or a trigger
Deferred with Automatic Query (depicted in second slide graphic)	Postpones potentially expensive detail query processing until the cursor visits the related blocks
Deferred without Automatic Query (depicted in first third graphic)	Allows entry of additional query criteria in the detail block before querying
Prevent Masterless Operations	Ensures that the detail block cannot be queried or used to insert records when a master record is not displayed

Note: In the New Relation dialog box, setting the Deferred property to Yes enables the Auto Query check box.

Modifying the Structure of a Data Block

You can modify the structure by using:

- **Reentrant Data Block Wizard:**
 1. Select frame or object in the Layout Editor, or data block or frame in Object Navigator.
 2. Select Tools > Data Block Wizard, or right-click and select Data Block Wizard, or click Data Block Wizard.
- **Object Navigator:**
 - Create or delete items.
 - Change item properties.
- **Block Property Palette:** Change property values.



Modifying the Structure of a Data Block

After you create a data block, you may want to customize or modify it by performing one of the following:

- Reenter the Data Block Wizard and use it to make the changes.
- Make manual changes, such as adding or deleting items, in Object Navigator.
- Change the property values of the block by using the Property Palette.

Invoking the Data Block Wizard in Reentrant Mode




A very powerful feature of the Data Block Wizard is its ability to operate in reentrant mode. Use reentrant mode to modify the data block, even if the block was not originally created with the Data Block Wizard.

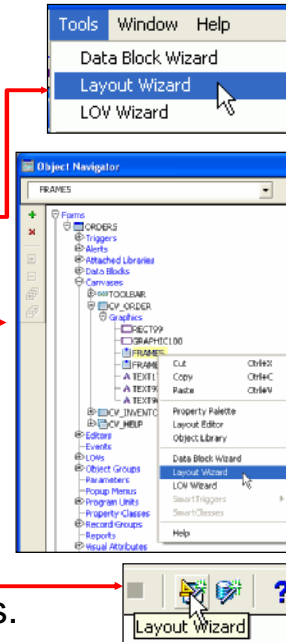
To invoke the Data Block Wizard in reentrant mode, perform the following steps:

1. Select the frame or a component of the block in either the Object Navigator or the Layout Editor.
2. Invoke the Data Block Wizard by performing one of the following steps, depicted in the screenshots:
 - Select Tools > Data Block Wizard from the menu.
 - Right-click and select Data Block Wizard from the pop-up menu.
 - Click Data Block Wizard.

Modifying the Layout of a Data Block

You can modify the layout by using:

- Reentrant Layout Wizard:
 - Select frame in Object Navigator or Layout Editor.
 - Select Tools > Layout Wizard,  or right-click and select Layout Wizard,  or click Layout Wizard.
- Layout Editor:
 - Select Tools > Layout Editor.
 - Make changes manually.
- Frame Property Palette: Change values. 



Modifying the Layout of a Data Block

You may want to customize or modify the layout of the data block items on the canvas. You can do this by performing one of the following steps:

- Reenter the Layout Wizard (see the next section) and use it to make the changes.
- Select Tools > Layout Editor to invoke the Layout Editor and make changes manually in the editor.
- Change the property values of the frame in its Property Palette.

Invoking the Layout Wizard in Reentrant Mode

A very powerful feature of the Layout Wizard is its ability to operate in reentrant mode. Use reentrant mode to modify the layout of items in an existing frame, even if the frame was not originally created with the Layout Wizard.

Modifying the Layout of a Data Block (continued)

You can invoke the Layout Wizard in reentrant mode from the Object Navigator or the Layout Editor:

1. Select the appropriate frame in the Object Navigator (under the Canvases node) or in the Layout Editor.
2. As shown in the screenshots, select Tools > Layout Wizard,
or
right-click the frame and select the Layout Wizard option,
or
if you are in the Layout Editor, click Layout Wizard.

Note: Before you reenter the Layout Wizard, it is important to select the correct frame in the Object Navigator or the Layout Editor. If you overlook this when you reenter the Layout Wizard, you may create an additional frame instead of modifying the current frame.

Either method takes you to the Data Block page in the Layout Wizard. Use Next and Back as you do when not in reentrant mode, or go directly to a certain page by clicking its tab.

Summary

In this lesson, you should have learned that:

- You can create data blocks with relationships by using the Data Block Wizard or by manually creating a Relation object
- When you run a master-detail form, block coordination is automatic depending on properties of the Relation object
- You can modify a data block manually or with the Data Block Wizard in reentrant mode
- You can modify the layout manually or with the Layout Wizard in reentrant mode

ORACLE

5 - 17

Copyright © 2009, Oracle. All rights reserved.

Summary

In this lesson, you should have learned about:

- Creating data blocks with a master-detail relationship
- Modifying the data block layout:
 - Using reentrant wizards
 - Changing frame properties

Practice 5: Overview

This practice covers the following topics:

- Creating a master-detail form module
- Modifying data block layout by using the Layout Wizard in reentrant mode
- Saving and running the form module

ORACLE

5 - 18

Copyright © 2009, Oracle. All rights reserved.

Practice 5: Overview

In this practice, you create a new form module that displays the master-detail information.

- Create a master-detail form module called Orders. Create a master block based on the `ORDERS` table and a detail block based on the `ITEMS` table. Create a third data block that is not related to any other block in the form module. Base this block on the `INVENTORIES` table, and manually create a relation with the block based on the item table. Use the Forms Builder wizards to create all three data blocks.
- Invoke the Layout Wizard in reentrant mode, and change the layout of the `ITEMS` and `INVENTORIES` data blocks.
- Save and run the new form module on the Web.

6

Working with Data Blocks and Frames

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Identify the components of the Property Palette
- Manage object properties
- Create and use Visual Attributes
- Control the behavior and appearance of data blocks
- Control frame properties
- Create blocks that do not directly correspond to database tables
- Delete data blocks and their components

ORACLE

6 - 2

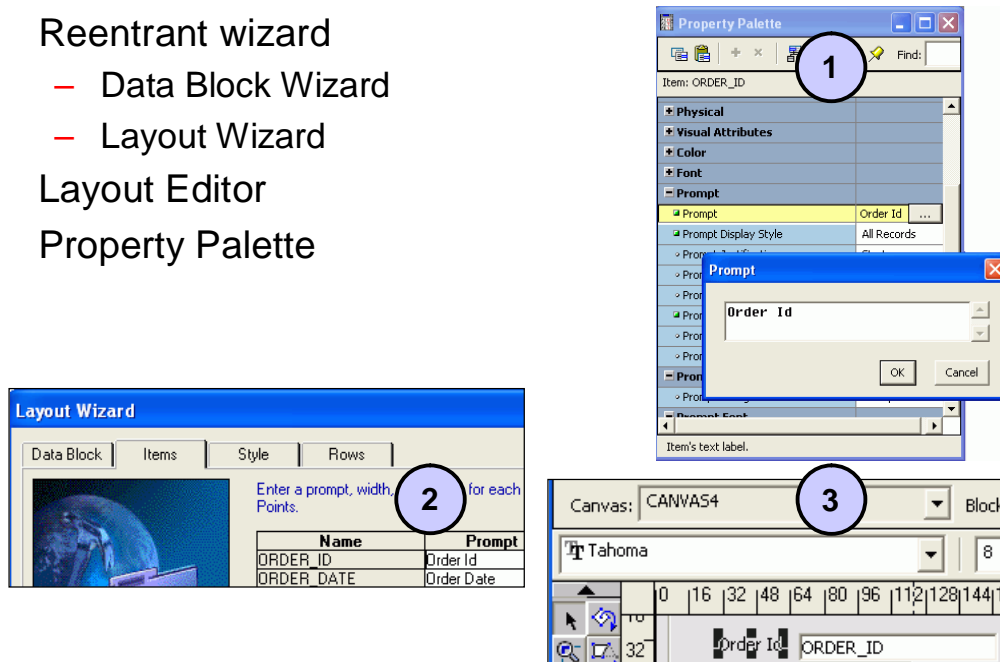
Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

In this lesson, you learn how to customize existing data blocks and modify frames. You also learn how to include blocks that are not associated with the database.

Managing Object Properties

- Reentrant wizard
 - Data Block Wizard
 - Layout Wizard
- Layout Editor
- Property Palette



Managing Object Properties

There are three ways to modify properties of Forms Builder objects:

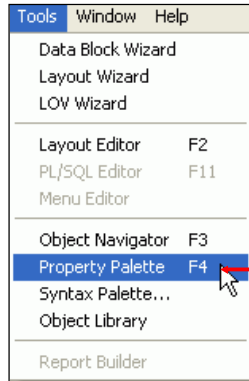
- **Reentrant Wizards:** You can modify data block and layout properties through the reentrant wizards, as explained in the previous lesson.
- **Layout Editor:** If the object appears on a canvas, you can modify properties by using the graphical Layout Editor.
- **Property Palette:** You can set individual properties for each Forms Builder object in its Property Palette.

The wizards, the Layout Editor, and the Property Palette all depict object properties. Changes made in one tool are reflected in the others. In the example in the slide, the prompt for Order_Id is shown identically in:

1. Property Palette
2. Reentrant layout wizard
3. Layout Editor

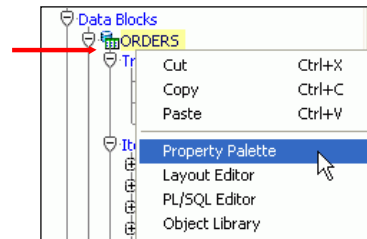
You can use the Property Palette to control the behavior and appearance of any Forms Builder object with a greater degree of granularity. In the Property Palette, you can fine-tune objects that you have initially created in the wizards or the Layout Editor.

Displaying the Property Palette



To display the Property Palette, use one of the following methods:

- Select Tools > Property Palette (or use the shortcut key).
- Double-click the object icon in the Object Navigator.
- Double-click the object in the Layout Editor.
- Right-click the object icon in the Object Navigator.
- Right-click the object in the Layout Editor.



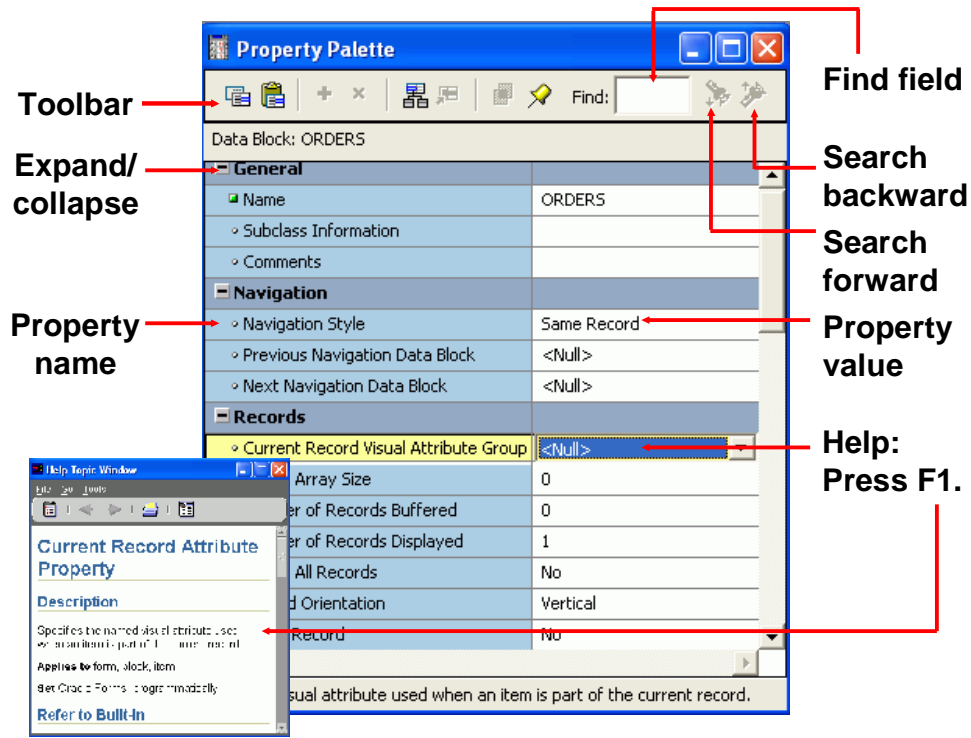
Displaying the Property Palette

Every object in a form module, as well as the form module itself, has properties that dictate the object's behavior. When an object is first created, it is automatically assigned several property values by default. You can change these property values in the Property Palette.

To display the Property Palette of an object, use one of the following methods:

- Select the object in the Object Navigator, and then select Tools > Property Palette from the menu, as shown in the first screenshot.
- Double-click the object icon for the object in the Object Navigator (except for code objects and canvases).
- Double-click an item in the Layout Editor.
- Right-click the object in the Layout Editor or the object icon in the Object Navigator. From the pop-up menu, select the Property Palette option, as shown in the second screenshot.

Property Palette: Features



ORACLE

6 - 5

Copyright © 2009, Oracle. All rights reserved.

Property Palette: Features

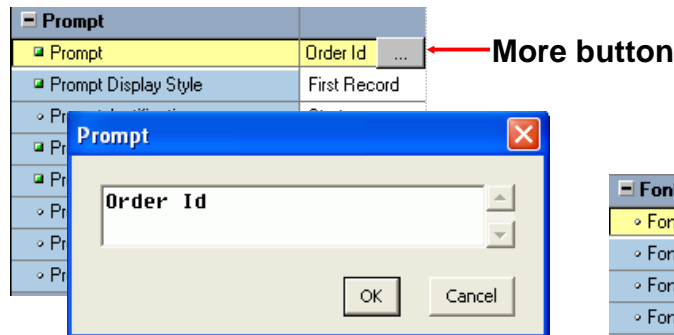
The following are the features of the Property Palette:

Feature	Description
Property list	Displays a two-column list of names and values of properties that are valid for a particular object. Properties are grouped under functional headings or nodes. You can expand or collapse a node by using the plus and minus icons beside the node name.
Find field	Enables you to quickly locate a particular property. The Search Forward and Search Backward buttons enhance your search.
Toolbar	Consists of a series of buttons that provide quick access to commands
Help	Enables you to obtain description, usage, and other information about any property by pressing F1 with the property selected

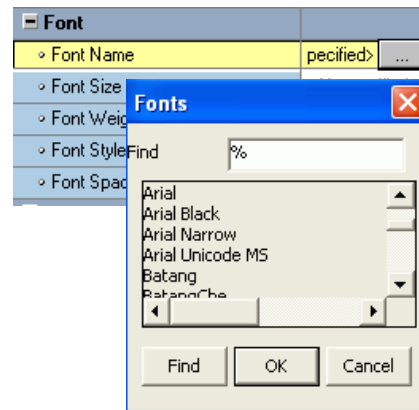
The screenshots show each of these features of the Property Palette and the Help window for a particular property.

Using the Property Palette

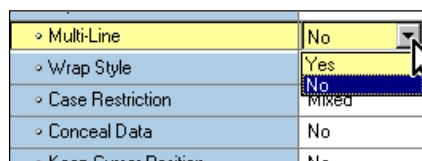
Text field



LOV window



Pop-up list







Using the Property Palette

Each form object has various types of properties. Properties are manipulated differently, depending on the property type. The following is a summary of the controls that are used in the Property Palette:

Property Control	Description
Text field	A Text field is displayed where the current property can be set by entering a text value. For longer text values, an iconic button also appears, enabling you to open a text editor, as shown in the top-left screenshot in the slide.
Pop-up list	This appears where a fixed set of values, such as Yes or No, is allowed for the property. Click the down arrow to open the list, as shown in the bottom-left screenshot, and then select a value.
LOV window	LOVs occur where a potentially large list of possible values is available. Click the button in the property value column to invoke an LOV, as shown in the screenshot at the right of the slide.
More button	Use this when more complex settings are needed. Click More to open the extra dialog box, similar to the example in the screenshot at the top-left of the slide.

Differentiating Property Palette Icons

Changed		Visual Attribute Group	VISUAL_A
Default		Prompt Visual Attribute Group	DEFAULT
Overridden		Foreground Color	magenta
Inherited		Background Color	gray





ORACLE

6 - 7

Copyright © 2009, Oracle. All rights reserved.

Differentiating Property Palette Icons

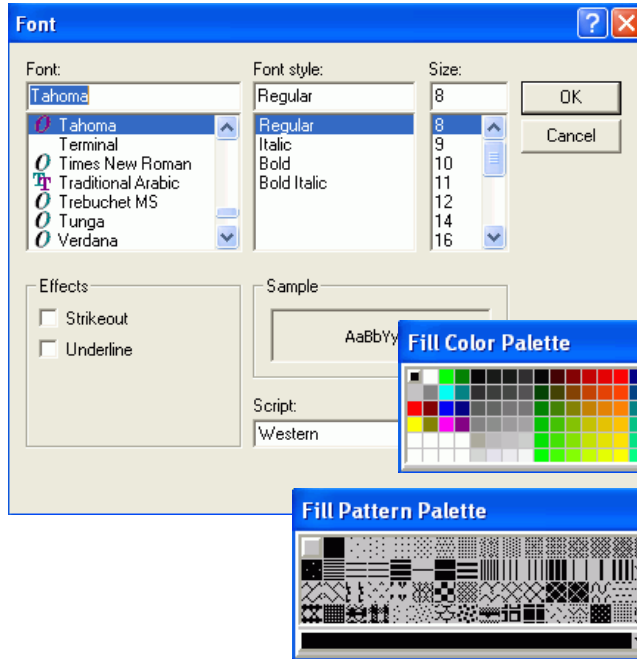
Each property in a Property Palette has an icon to its left. The following is a summary of these icons and their description:

Icon	Description
 Circle	Specifies that the property value is the default value
 Square	Specifies that the property value has been changed from the default
 Arrow	Specifies that the property value is inherited
 Arrow with a cross	Specifies that the property value was inherited but has been overridden

Note: After you activate the Property Palette, its window remains open until you close it. The window automatically displays the properties of each object you visit in the Layout Editor or the Object Navigator. This is because, by default, the list of properties in the Property Palette is synchronized whenever you select an object.

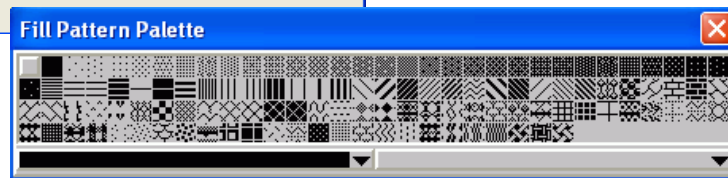
You can turn the synchronization on or off for a specific palette by clicking Pin/Unpin on the Property Palette toolbar.

Visual Attributes: Overview



A visual attribute is a named set of properties defining:

- Font
- Color
- Pattern



Visual Attributes: Overview

Visual attributes are the font, color, and pattern properties that you set for form and menu objects. The screenshots show dialog boxes for setting font, fill color, and fill pattern.

You can create a named set of such properties; this named set is called a Visual Attribute, which is another object that you can create in the Object Navigator with properties such as font, color, and pattern combinations. Set the Visual Attribute Type property of the Visual Attribute to Title if you plan to apply it to objects such as frame titles, or to Prompt if it is used for prompts. Otherwise, set Visual Attribute Type to Common, which is the default.

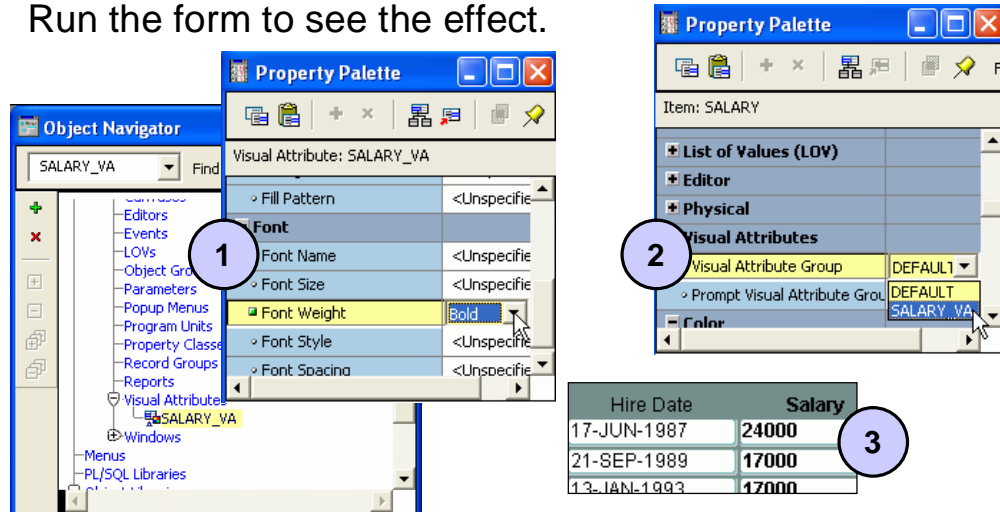
Every interface object in a Forms application has a property called Visual Attribute Group, which determines how the individual Visual Attribute settings of an object are derived. The Visual Attribute Group property can be set to Default, NULL, or the name of a Visual Attribute object. Blocks have a Current Record Visual Attribute Group property that defines the Visual Attribute to be used for the current record in the block.

Partial Visual Attributes

You can define a Visual Attribute by setting only the properties that you want to be inherited by the objects that use them. This means that you can apply a Visual Attribute that changes the font color without having to set the font name.

Using Visual Attributes

1. Create a Visual Attribute.
2. Set the Visual Attribute–related property of an object to the desired Visual Attribute.
3. Run the form to see the effect.



6 - 9

Copyright © 2009, Oracle. All rights reserved.

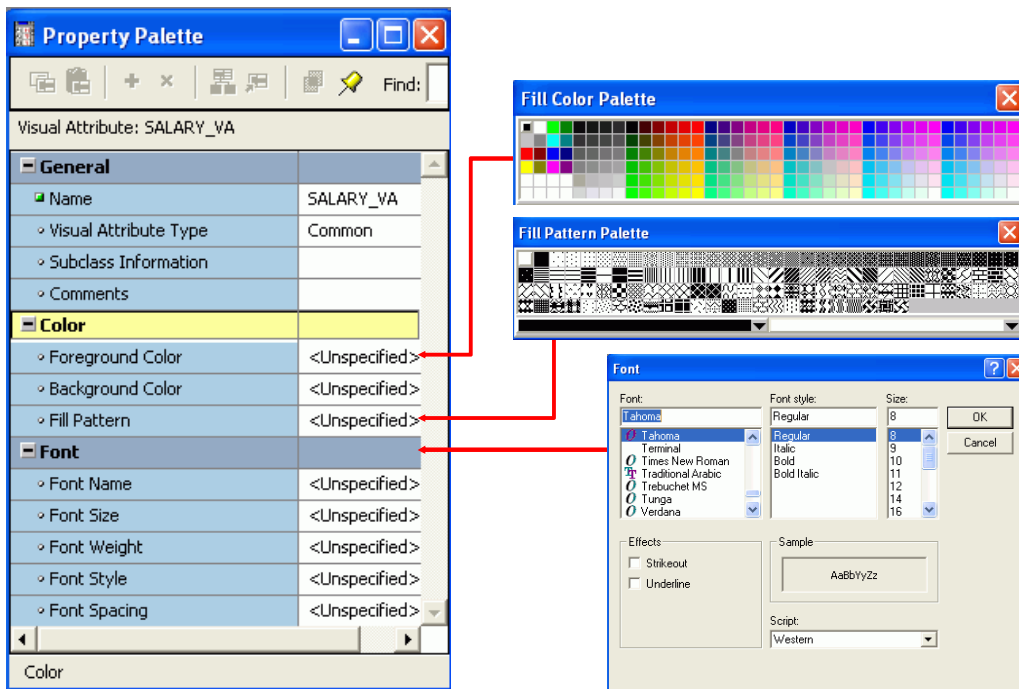
ORACLE

Using Visual Attributes

To use a Visual Attribute, perform the following steps:

1. Create the Visual Attribute:
 - Select the Visual Attributes node in the Object Navigator.
 - Click Create.
 - Invoke the Property Palette for the Visual Attribute and set the desired font, color, and pattern properties. You need set only those properties that you want to use. The screenshot at the left shows defining a Visual Attribute named SALARY_VA with a font weight of bold.
2. Set the object properties to use the new Visual Attribute:
 - For items and canvases, set the Visual Attribute Group property. The top-right screenshot shows setting the Visual Attribute Group property for the Salary text item to SALARY_VA, the Visual Attribute that was defined.
 - For blocks, you can set the Current Record Visual Attribute Group property.
3. Run the form to see the changes. The bottom-right screenshot shows that at run time, the Salary text item appears in a bold font.

Using Font, Pattern, and Color Pickers



ORACLE

6 - 10

Copyright © 2009, Oracle. All rights reserved.

Using Font, Pattern, and Color Pickers

When you create Visual Attributes, you can use the Font, Pattern, and Color pickers to select the font, pattern, and color.

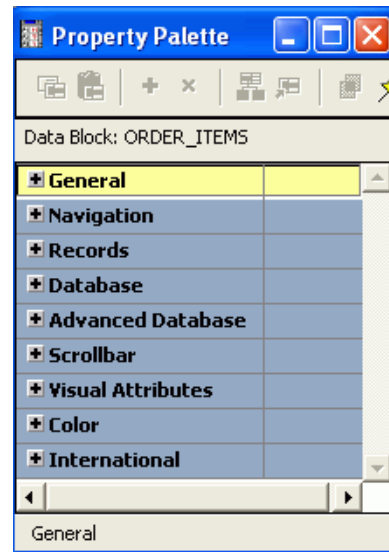
When changing a font from the Property Palette, you can click the Font group itself to invoke the Font Picker. This enables you to select all font properties from one window; otherwise, you can select each font property and invoke individual windows or pop-up lists that are specific to that property.

The screenshots show the fill color picker, the pattern picker, and the font picker that you can invoke from the Property Palette. When you click the value of one of the visual properties, such as Foreground Color, Fill Pattern, or Font, a More button appears that can invoke the appropriate interface for setting the property.

Controlling Data Block Behavior and Appearance

Data Block Property groups:

- General
- Navigation
- Records
- Database
- Advanced Database
- Scrollbar
- Visual Attributes
- Color
- International



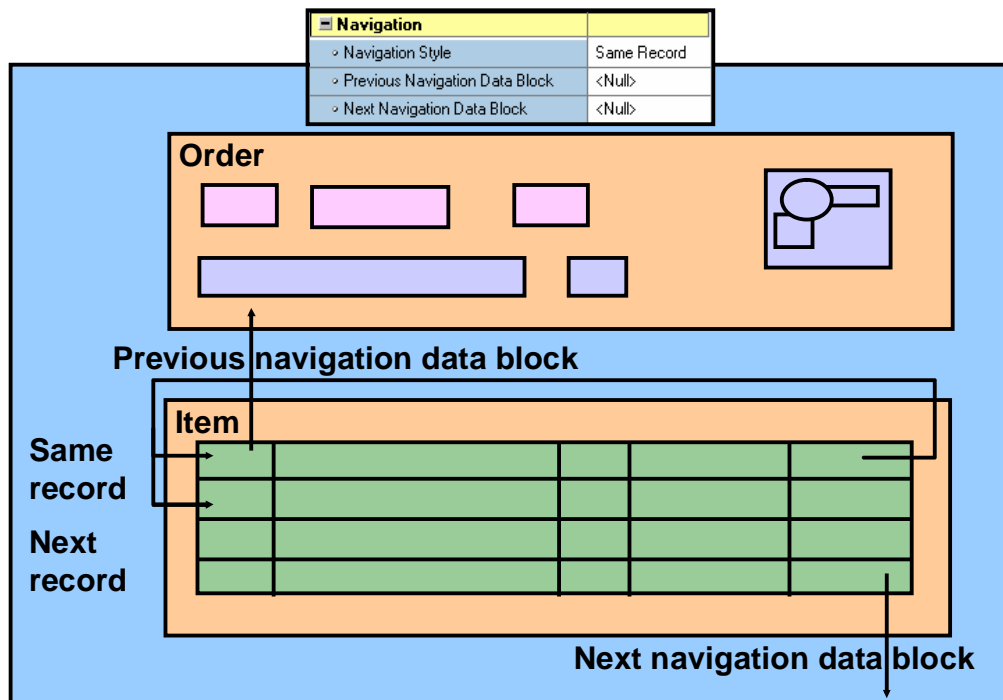
ORACLE

Controlling Data Block Behavior and Appearance

Each data block has several properties. These properties are divided into groups as shown in the slide.

You can modify the properties of the data block to control both the appearance and the behavior of the block.

Setting a Block's Navigation Properties

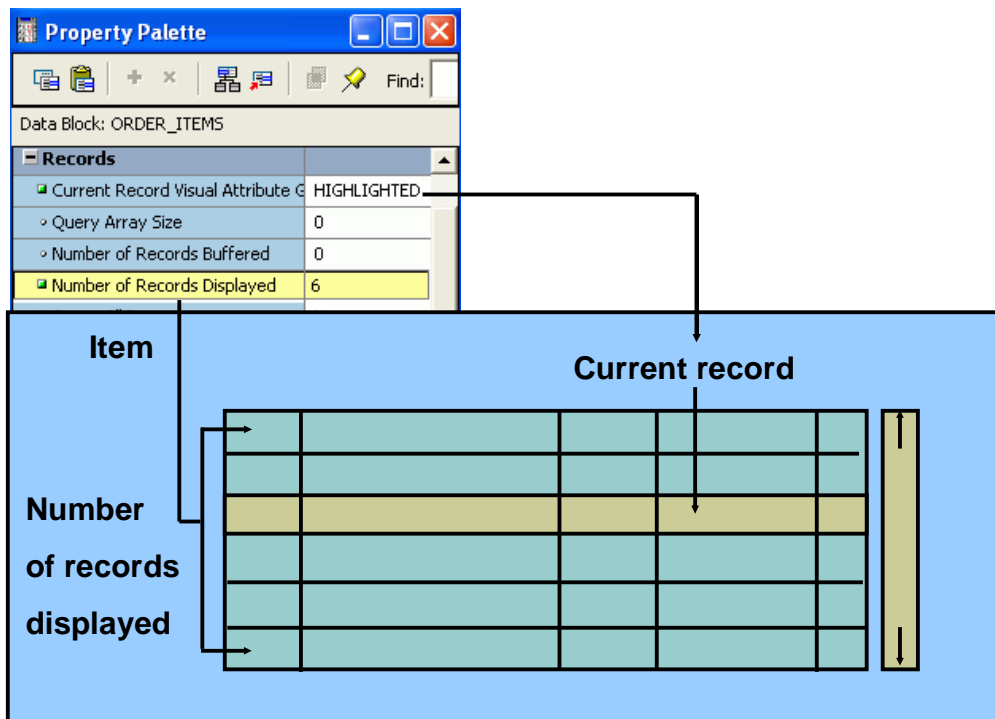


Setting a Block's Navigation Properties

The graphics show that, by default, when you navigate beyond the last item in a record, Forms returns you to the beginning of the same record. When you perform an operation to move to the previous or next data block at run time, Forms moves control to the previous or next adjacent data block in the sequence that these blocks appear in the Object Navigator. The screenshot shows the navigation properties that you can use to change default behavior:

- **Navigation Style:** With this property, you can change this behavior to navigate to the next record or data block instead. The valid settings are Same Record (default), Change Record, or Change Data Block. For example, if you want the cursor to move to the next record when you reach the end of the current record, set the Navigation Style property for the block to Change Record.
- **Previous/Next Navigation Data Block:** These properties enable you to name the previous or next data block, rather than using the default Object Navigator sequence.

Setting a Block's Records Properties



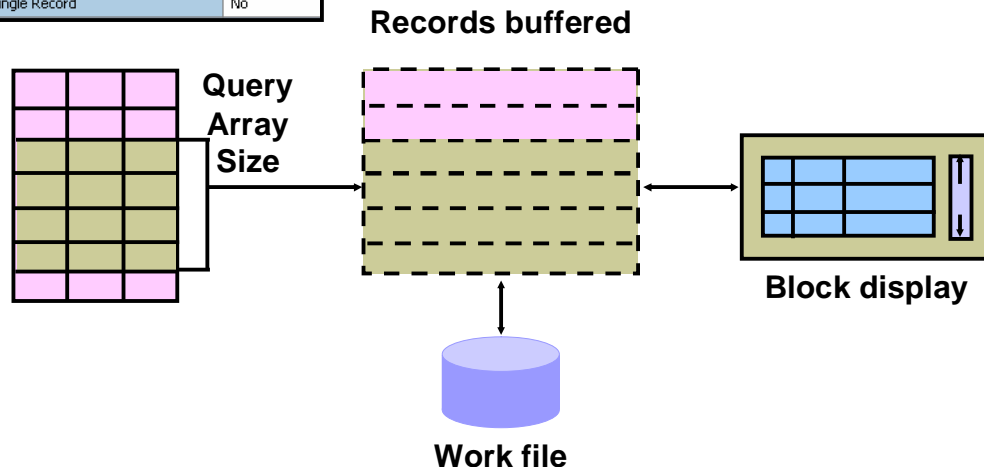
Setting a Block's Records Properties

The screenshot shows the properties of a block that pertain to records:

- **Current Record Visual Attribute Group:** This is the Visual Attribute that is used to highlight the current record in the data block.
- **Number of Records Displayed:** This property specifies the maximum number of records that the data block can display on the canvas at one time and how many records you can see at once. If you change this value, make sure there is enough room on the canvas layout for the number of records, or the objects may overlap. The slide shows this property set to 6, with 6 rows displayed at run time.

Setting a Block's Records Properties

Records	
Current Record Visual Attribute Group	<Null>
Query Array Size	4
Number of Records Buffered	0
Number of Records Displayed	3
Query All Records	No
Record Orientation	Vertical
Single Record	No



ORACLE

6 - 14

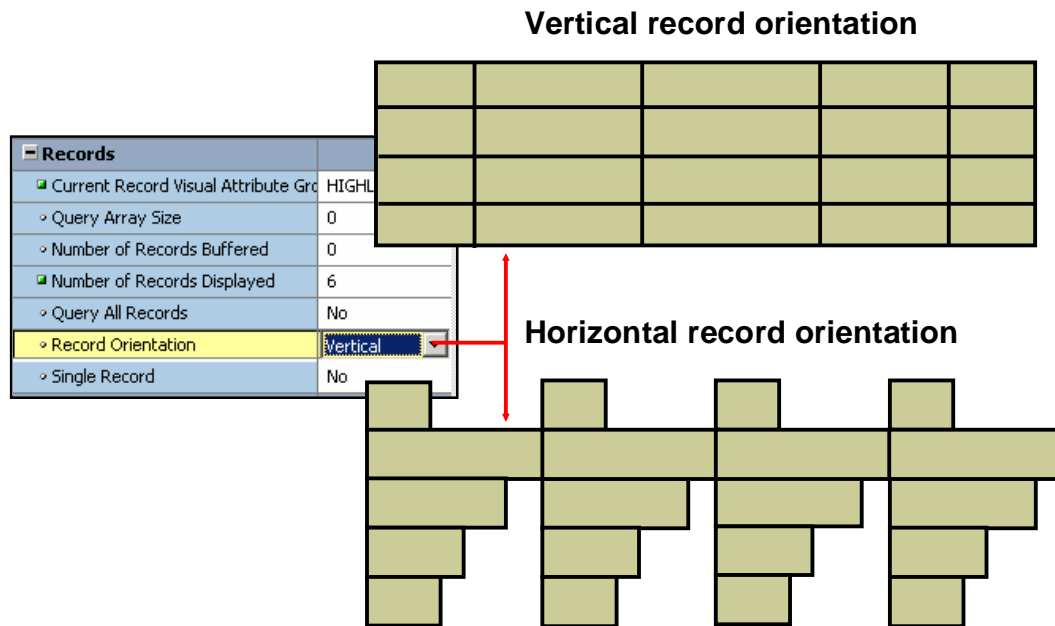
Copyright © 2009, Oracle. All rights reserved.

Setting a Block's Records Properties (continued)

- **Query Array Size:** This specifies the maximum number of records that Forms should fetch from the database at one time. A lower value in this property value means faster response time; however, a larger value means fewer calls to the database for records, thereby resulting in reduced overall processing time. When set to 0, this property defaults to the Number of Records Displayed.
- **Number of Records Buffered:** This is the minimum amount of buffer space retained for holding queried records in the data block. The minimum setting allowed is the value of the Number of Records Displayed property plus 3. Forms buffers any additional records to a temporary disk file. A higher value improves processing speed, but uses more memory.

The screenshot shows Query Array Size set to 4 and Number of Records Buffered set to 0. The result of these settings is that 4 records at a time are fetched by the query into the record buffer, whose size, since Number of Records Buffered is 0, is 6 (the minimum size of Number of Records Displayed plus 3.)

Setting a Block's Records Properties



ORACLE

Setting a Block's Records Properties (continued)

- **Query All Records:** This property specifies whether all the records matching the query criteria should be fetched when a query is executed. (This property is necessary to support the Calculated Field feature.)
- **Record Orientation:** This property determines the orientation of records in the data block—horizontal or vertical. When you set this property, Forms Builder adjusts the display position of items in the data block accordingly. The top graphic shows vertical record orientation, with records displayed with each record below the previous one, while the lower graphic shows horizontal orientation, with records displayed beside one another.
- **Single Record:** This property specifies that the control block should always contain one record. Set this property to Yes for a control block that contains a summary calculated item. You cannot set this property to Yes for a data block.

Setting a Block's Database Properties

Use properties in the Database group to control:

- Type of block—data or control block
- Query, insert, update, and delete operations on the data block
- Data block's data source
- Query search criteria and default sort order
- Maximum query time
- Maximum number of records fetched

Database	
Database Data Block	Yes
Enforce Primary Key	No
Query Allowed	Yes
Query Data Source Type	Table
Query Data Source Name	ORDERS
Query Data Source Columns	
Query Data Source Arguments	
Alias	
Include REF Item	No
WHERE Clause	
ORDER BY Clause	order_id
Optimizer Hint	
Insert Allowed	Yes
Update Allowed	Yes
Locking Mode	Automatic
Delete Allowed	Yes
Key Mode	Automatic
Update Changed Columns Only	No
Enforce Column Security	No
Maximum Query Time	0
Maximum Records Fetched	0

ORACLE

Setting a Block's Database Properties

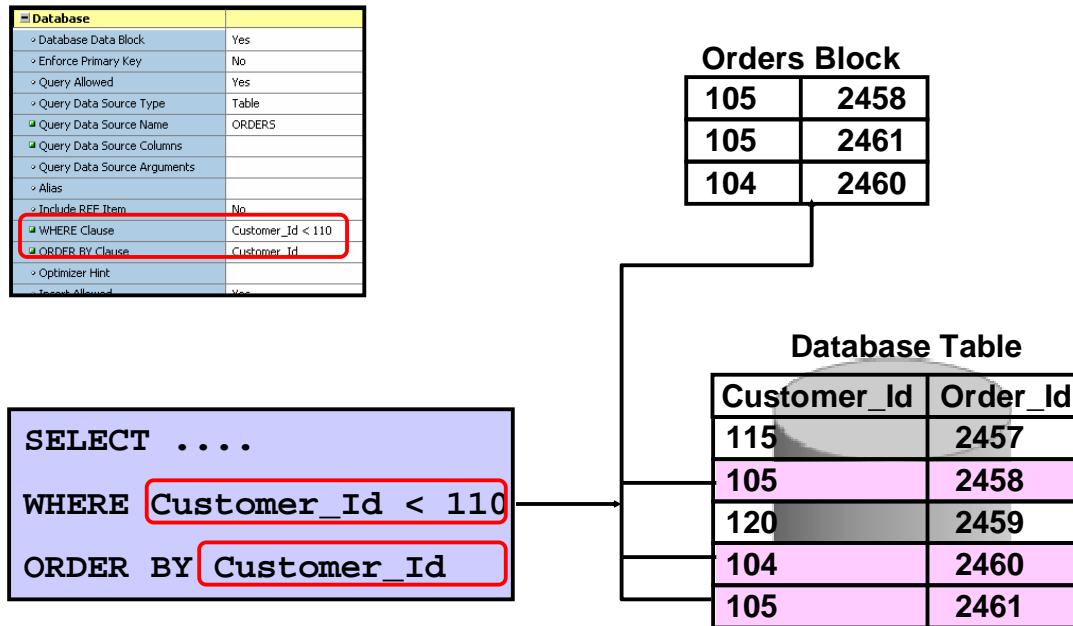
In the Database group of the Property Palette, you can set numerous properties to control interaction with the database server, as depicted in the screenshot. Some of these properties are:

- **Database Data Block:** Set to Yes if the data block is based on a database object, and No if it is a control block
- **Enforce Primary Key:** Controls whether Forms checks for uniqueness before inserting or updating records in the base table, in order to avoid committing duplicate rows in the database. A value of Yes means that Forms checks that inserted or updated records are unique before an attempt is made to commit possible duplicate rows.
- **Query Allowed/Insert Allowed/Update Allowed/Delete Allowed:** Control whether the associated operations can be performed on the data block records
- **Query Data Source Type:** Specifies the type of the query data source for the data block. Possible values for this property are None, Table, Procedure, Transactional Triggers, or the FROM clause query.
- **Query Data Source Name:** Specifies the name of the query data source for the data block. This property is used only if the type of the query data source is Table, the FROM clause query, or Procedure.

Setting a Block's Database Properties (continued)

- **Query Data Source Columns:** Specifies in a dialog box, the name and data type of the columns associated with the query data source. This property is used only if the type of the query data source is Table, the FROM clause query, or Procedure.
- **Query Data Source Arguments:** Specifies in a dialog box, the names, data types, and values of the arguments that are to be passed to the procedure for querying data. This property is valid only when the Query Data Source Type property is set to Procedure.
- **Optimizer Hint:** Specifies a hint string that Forms passes to the Optimizer when constructing implicit SQL on the data block. The Optimizer can improve the performance of database transactions.
- **Locking Mode/Key Mode:** Controls how Forms handles records and transactions when the data block is primarily associated with non-Oracle data sources. The default settings are usually appropriate for data blocks connected with an Oracle database.
- **Update Changed Columns Only:** By default, this property value is set to No, so that all columns are included in the default UPDATE statement; Forms can reuse the same SQL statement for multiple updates without the database having to reparse each time in its System Global Area (SGA). When this property is set to Yes, only those items updated by the operator are written to their corresponding database columns. If the operator commonly updates or inserts records with only one or two columns, this can save network traffic. However, the database must reparse the UPDATE statement each time, which can degrade performance.
- **Enforce Column Security:** When this property is set to Yes, items in the data block can be updated only if the current user has privileges to update the corresponding database columns.
- **Maximum Query Time:** Provides the option to abort a query when the elapsed time of the query exceeds the value of this property; useful when the Query All Records property is set to Yes
- **Maximum Records Fetched:** Provides the option to abort a query when the number of records fetched exceeds the value of this property; useful when the Query All Records property is set to Yes

Setting a Block's Database Properties: Query Clauses



ORACLE

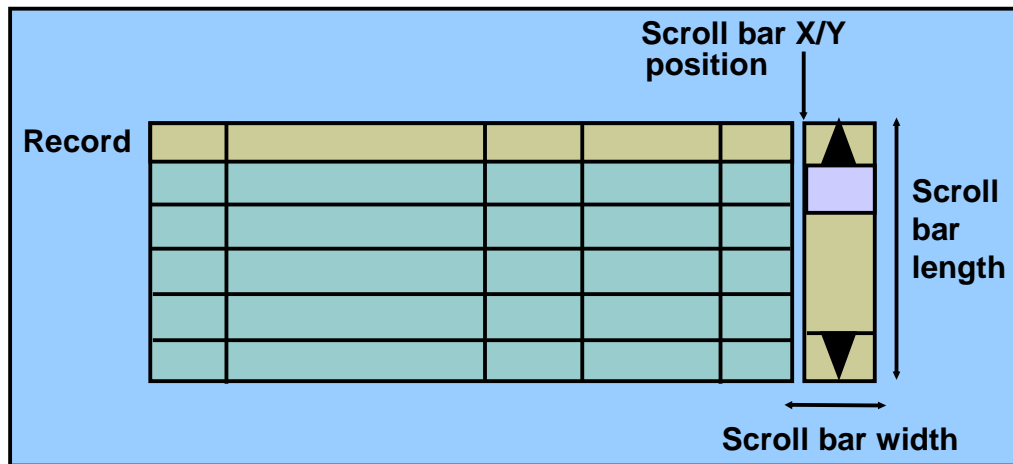
Setting a Block's Database Properties: Query Clauses

There are two database properties that affect how the block's data query is constructed:

- **WHERE Clause:** Specifies a SQL condition that is attached to every default SELECT statement associated with the data block through implicit SQL; use to define general restrictions on the rows that this data block may fetch. This clause is automatically appended (ANDed) with any conditions supplied by the operator in Enter Query mode.
- **ORDER BY Clause:** Defines a default order for records displayed from a query. The operator can alter this order by using the Query/Where dialog box at run time.

The graphics show how the block's WHERE Clause and ORDER BY clause are used in the WHERE clause and the ORDER BY clause of the SQL statement that is sent to the database. The WHERE clause is appended to any existing query criteria (in this case there are none) to further restrict the rows returned by the query. The ORDER BY clause determines the display order in the data block.

Setting a Block's Scroll Bar Properties



ORACLE

6 - 19

Copyright © 2009, Oracle. All rights reserved.

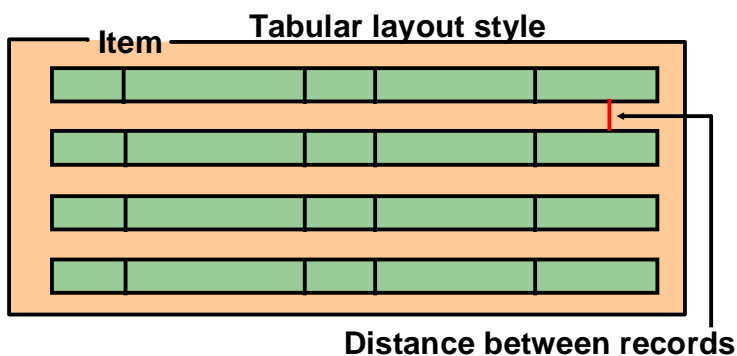
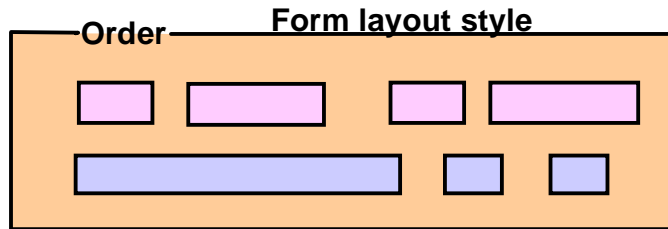
Setting a Block's Scroll Bar Properties

In the Scrollbar group of the Property Palette, you can set numerous properties to the appearance and function of the data block's scroll bar. Some of these properties are:

- **Show Scroll Bar:** Specifies whether Forms Builder should create a scroll bar for the data block. To delete an existing scroll bar, set this property to No.
- **Scroll Bar Canvas/Tab Page:** Specifies the canvas and tab page on which the data block scroll bar is displayed. The specified canvas or tab page must exist in the form.
- **Scroll Bar Orientation:** Specifies whether the scroll bar should be displayed horizontally or vertically
- **Scroll Bar X/Y Position:** Specifies the x and y coordinates (measured in the coordination system units of the form) where the scroll bar displays on the canvas. The default value for both the coordinates is 0. These properties are shown in the slide.
- **Scroll Bar Width/Length:** Specifies the width and height of the scroll bar, as shown in the slide
- **Reverse Direction:** Determines whether the scroll bar should scroll in reverse

Controlling Frame Properties

Graphics: FRAMES	
General	
Layout Frame	
Layout Data Block	ORDERS
Update Layout	Automatically
Layout Style	Tabular
Frame Alignment	Column
Single Object Alignment	Start
Horizontal Margin	5
Vertical Margin	14
Horizontal Object Offset	0
Vertical Object Offset	0
Allow Expansion	Yes
Shrinkwrap	No
Vertical Fill	Yes
Maximum Objects per Line	0
Start Prompt Alignment	Start
Start Prompt Offset	5
Top Prompt Alignment	Center
Top Prompt Offset	0
Allow Multi-line Prompts	Yes
Allow Top-Attached Prompts	No
Allow Start-Attached Prompts	No
Records	
Number of Records Displayed	6
Distance Between Records	10



ORACLE

Controlling Frame Properties

The selections that you make in the Layout Wizard when creating a data block are recorded as properties of the resulting layout frame object. You can change frame properties to modify the arrangements of items within a data block. The main frame properties are as follows:

- **Layout Data Block:** Specifies the name of the data block with which the frame is associated. The items within this data block are arranged within the frame. A data block can be associated with only one frame. You cannot arrange a block item within multiple frames.
- **Update Layout:** Specifies when the frame layout is updated. Valid settings are:
 - **Automatically:** The layout is updated whenever you move or resize the frame, or modify any frame layout property.
 - **Manually:** The layout is updated whenever you use the Layout Wizard to modify the frame, or in the Layout Editor, when you click Update Layout or select the Layout > Update Layout menu option.
 - **Locked:** The layout is locked and cannot be updated.
- **Layout Style:** Specifies the layout style for the items within the frame. Choose between Form and Tabular styles.

Controlling Frame Properties (continued)

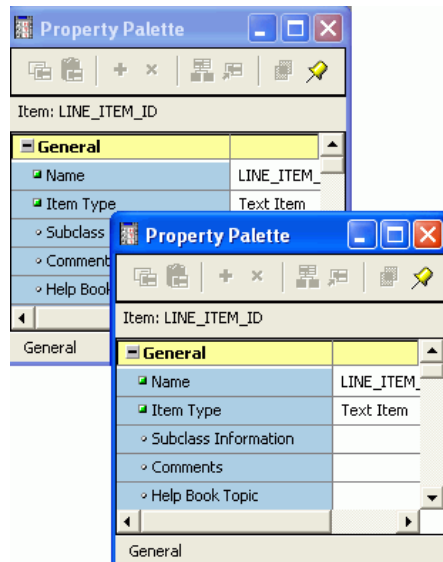
- **Distance Between Records:** Specifies the physical distance (measured in the form's coordination system units) with which to separate records displayed in the frame
- **X/Y Position:** Specifies the x and y coordinates (measured in the form's coordination system units) of the frame's position on the canvas
- **Width/Height:** Specifies the width and height of the frame (measured in the form's coordination system units)

The screenshot shows the Property Palette for a frame, and the graphics illustrate two of the frame's properties. The top graphic shows a form layout style, while the bottom one shows a tabular layout style. The size of the space between the records of a multirecord block is determined by the Distance Between Records property.

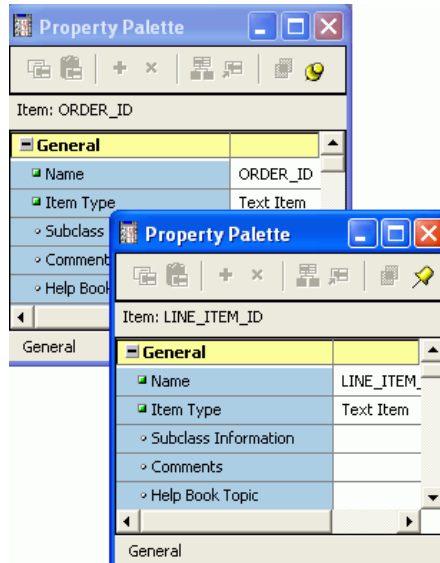
Note: You can arrange a frame as well as the objects within it manually in the Layout Editor.

Displaying Multiple Property Palettes

Two palettes for one item:



Two palettes for two items:



ORACLE

6 - 22

Copyright © 2009, Oracle. All rights reserved.

Displaying Multiple Property Palettes

More Than One Property Palette for One Object

You may want to see more properties for an object than there is room for in a single Property Palette. To display the properties of an object in multiple Property Palettes, as shown by the screenshots to the left side of the slide, perform the following steps:

1. Open a Property Palette for the object.
2. Press and hold the Shift key and double-click the object icon for the object in the Object Navigator or the Layout Editor.

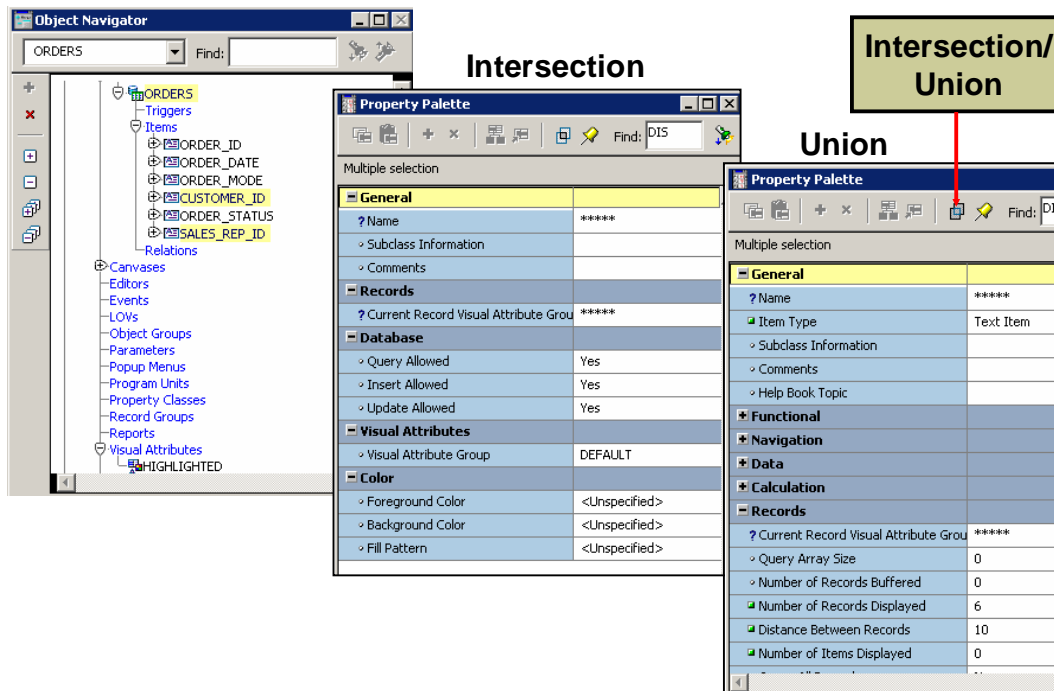
More Than One Property Palette for Multiple Objects

You may want to display properties for multiple objects simultaneously. To display the Property Palettes for multiple objects at the same time, as in the screen shots to the right side of the slide, perform the following steps:

1. Open the Property Palette of the first object.
2. Click Pin/Unpin on the toolbar to “freeze” this palette.
3. Invoke the Property Palette for another object. This Property Palette appears in a separate window.



If the second window is on top of the first one, drag it so that both windows are visible.

Setting Properties on Multiple Objects



Setting Properties on Multiple Objects

You can view and set the properties of several objects simultaneously, whether they are the same or different object types. You can select the objects in the Object Navigator and display a combination of the properties in the Property Palette. The combination may be:

-  **Intersection:** A subset in which you display only the common properties of the selected objects (This is the default set operator.)
-  **Union:** A superset in which you display both the common properties and the unique properties of the selected objects

Where there are differing values for a property across the selected objects, you see ***** in the property value. This changes to a definitive value after you enter a new value in the Property Palette. This new value then applies to each of the selected objects to which the property is relevant.

The screenshot to the left side of the slide shows selecting several objects, while the other two screenshots show how the Property Palette appears for an intersection (middle screenshot) and for a union (right screenshot) of properties. Note that two of the properties show a line of asterisks for the value, meaning that the selected objects have different values for those properties.

Setting Properties on Multiple Objects (continued)

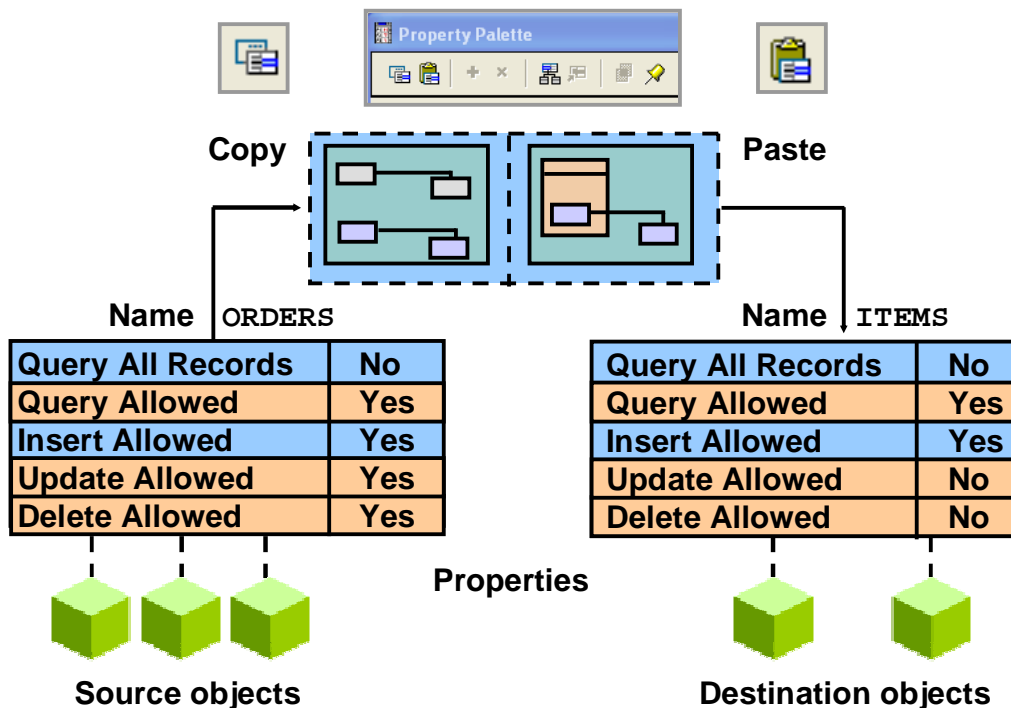
How to Set Properties on Multiple Objects

To set properties on multiple objects at one time, perform the following steps:

1. Open the Property Palette for one of the objects.
2. Press and hold the Ctrl key and select each object in the Object Navigator or the editors whose properties are to be viewed or changed in combination. The selected objects are highlighted.
3. Set the Intersection/Union button from the toolbar in the Property Palette to the desired set type. This button toggles between the two options.
4. Change the displayed properties, as required. Your changes are applied to all selected objects with these properties.

Note: With a union, you may see some properties that are not relevant to all of the selected objects. Changes to a property are applied only to objects that have the property.

Copying Properties



ORACLE

6 - 25

Copyright © 2009, Oracle. All rights reserved.

Copying Properties

You can copy the properties and values from the Property Palette to a buffer, so that they can be applied (pasted) to other objects in your design session. To copy properties, perform the following steps:

1. In the Property Palette, display and set the properties that are to be copied. This may be from one object or a combination of objects.
2. Select the properties to be copied:
 - To copy all property settings from the Property Palette, select Edit > Select All.
 - To copy the selected property settings only, press and hold Ctrl and select each property individually. This graphics at the left of the slide shows two properties being copied.
3. Click Copy Properties on the toolbar of the Property Palette.
4. From the Object Navigator, select the object into which the properties are to be copied.
5. In the Property Palette, click Paste Properties. The selected object receives values from all copied properties that are relevant to their object types, as shown in the graphics to the right side of the slide, where the pasted property values are applied.

Note: It is possible to copy the property settings of an object to objects of different types. In this case, properties that do not apply to the target object are ignored.

Copying Properties (continued)

Property Classes

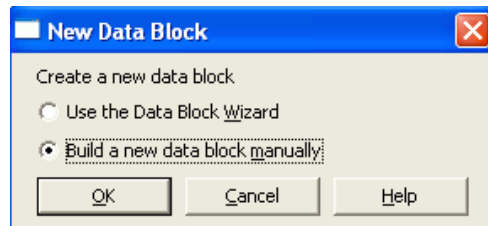
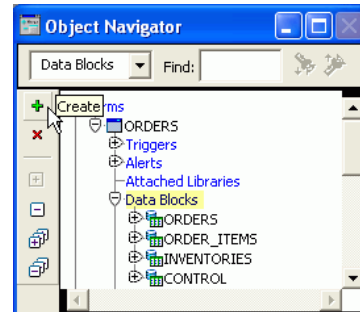
When you display a list of properties (from either one object or a combination of objects) in the Property Palette, the list of property names and associated values can be saved for future application to other objects. This is known as a property class, which is a Forms Builder object in its own right.

Objects can inherit some of their properties from a linked property class, so their properties change automatically if the associated properties are changed in the property class.

Property classes are discussed in more detail in the lesson titled “Sharing Objects and Code”.

Creating a Control Block

- Click the Data Blocks node.
- Click the Create icon.
or
Select Edit > Create.
- Select the Build a new data block manually option in the New Data Block dialog box.



Creating a Control Block

A control block is a block that is not associated with any database, and its items do not relate to any columns within any database table.

This means that Forms does not perform an automatic query when the operator issues an Enter Query or Execute Query command, nor does it issue an automatic Insert, Update, or Delete for the block when the operator saves changes to the database.

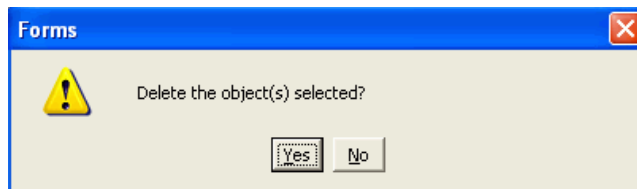
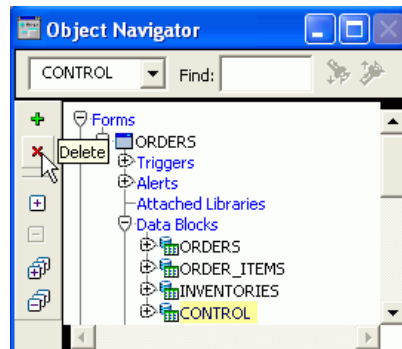
How to Create a Control Block

1. Click the Data Blocks node in the Object Navigator.
2. Click the Create icon on the toolbar, or select Edit > Create from the menu.
3. In the New Data Block dialog box, select the Build a new data block manually option.
4. Open the Property Palette of the new block and change its name.

Note: Because there are no database columns on which to base control block items, a control block has no items until you manually add them later.

Deleting a Data Block

- Select a data block for deletion.
- Click the Delete icon.
or
Press Delete.
- Click Yes in the alert box.



Deleting a Data Block

To delete a block, perform the following steps:

1. Select the block to be deleted in the Object Navigator.
2. Click the Delete icon on the toolbar, or press Delete.
3. An alert is displayed for delete confirmation, as shown in the screen shot at the bottom left of the slide. Click Yes to delete the block.

Note: Deleting a block also deletes its subordinate objects (items and triggers). If the block was a master or detail block in a relation, the relation is also deleted. However, the frame border and its title will remain. Delete the frame manually in the Layout Editor.

Summary

In this lesson, you should have learned that:

- The Property Palette:
 - Contains property names and values that enable you to modify Forms objects
 - Has tools to search for properties, inherit properties, expand or collapse property categories, and pop-up lists and dialog boxes for various properties
 - Shows different icons for default, changed, inherited, and overridden properties
- Block properties control the behavior and appearance of data blocks
- Frame properties control how block items are arranged
- You can create blocks that do not directly correspond to database tables by choosing to create the block manually rather than using the Data Block Wizard
- Deleting a block deletes all of its components

ORACLE

6 - 29

Copyright © 2009, Oracle. All rights reserved.

Summary

- Modify the data block properties in its Property Palette to change its behavior at run time.
- Data blocks have Navigation, Database, Records, Scrollbar, and other properties.
- Database properties include WHERE Clause, ORDER BY Clause, Query Data Source Type, and Maximum Records Fetched.
- You can change frame properties to modify the arrangements of items within a data block.
- You can copy properties between data blocks and other objects.
- You can view and change the properties of several objects together. You can use Intersection or Union settings to connect their properties in the Property Palette.

Practice 6: Overview

This practice covers the following topics:

- Modifying data block properties
- Modifying frame properties
- Creating a Visual Attribute
- Creating a control block

ORACLE

6 - 30

Copyright © 2009, Oracle. All rights reserved.

Practice 6: Overview

In this practice, you will perform the following tasks:

- Change properties in the CUSTOMERS data block and frame to change run-time appearance and behavior and design-time behavior of the frame.
- Change properties in the ITEMS and INVENTORIES data blocks to change their run-time appearance and behavior.
- Change the frame properties of all the data blocks in the ORDERS form to change their run-time appearance and to keep any layout changes you make manually in the Layout Editor.
- Create a Visual Attribute in the ORDERS form and use it to highlight the current record in the ITEMS and INVENTORIES data blocks at run time. Use the multiple selection feature in the Property Palette.
- Create a control block in the ORDERS form.
- Create a control block in the CUSTOMERS form.
- Save and run the forms after the changes are applied.



Working with Text Items

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe text items
- Create a text item
- Modify the appearance of a text item
- Control the data in a text item
- Alter the navigational behavior of a text item
- Enhance the relationship between the text item and the database
- Add functionality to a text item
- Display helpful messages

ORACLE

7 - 2

Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

The default item type in an Oracle Forms application is the text item or field. You have seen how creating a new data block based on a table creates text items for each selected column from that table. This lesson shows you how to customize text items to change their appearance and behavior.

Text Item: Overview

What is a text item?

- Default item type
- Interface object for:
 - Querying
 - Inserting
 - Updating
 - Deleting
- Behavior defined in the Property Palette



ORACLE

7 - 3

Copyright © 2009, Oracle. All rights reserved.

Text Item: Overview

A text item is an interface object with which you can query, insert, update, and delete data. A text item usually corresponds to a column in the database table. When an item is first created, its default type is text.

The item type determines the properties available in the Property Palette. In this lesson, you look at the properties of a text item. The remaining item types are covered in subsequent lessons.

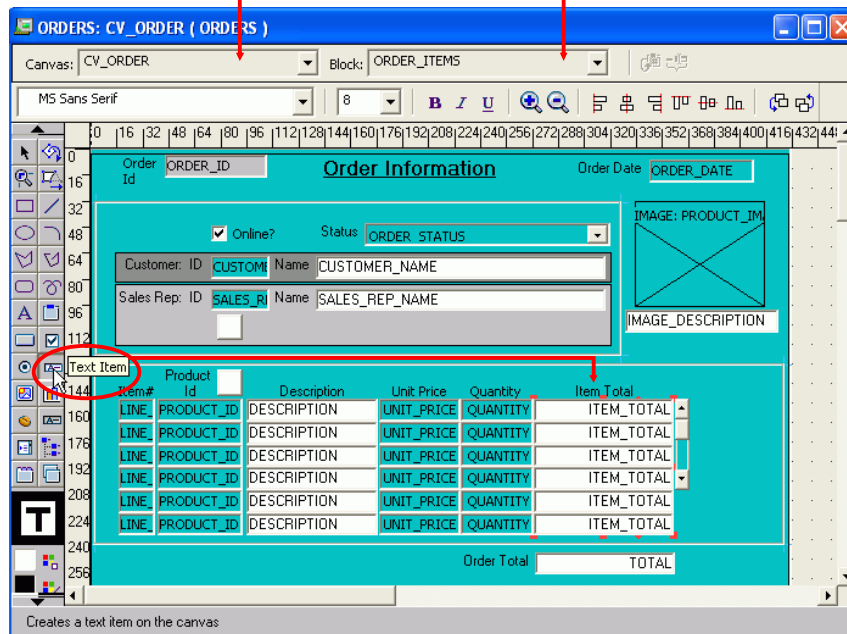
Use the Property Palette to define, alter, or examine the characteristics of items.

The screenshot in the slide shows the tool for creating a text item in the Layout Editor.

Creating a Text Item

Canvas selection

Block selection



ORACLE

7 - 4

Copyright © 2009, Oracle. All rights reserved.

Creating a Text Item

You can create a text item by doing one of the following:

- Converting an existing item into a text item
- Using the Text Item tool in the Layout Editor
- Using the Create icon in the Object Navigator
- Using the wizards

How to Create a Text Item in the Layout Editor

1. Invoke the Layout Editor, as shown in the screenshot.
It is important to point to the correct data block where you want to create the text item. In the Layout Editor, select the data block from the Block pop-up list.
2. Click the Text Item tool.
3. Click the canvas.
The text item appears.
4. Double-click the text item.
The text item Property Palette appears.
5. Set the item properties as required.

Creating a Text Item (continued)

How to Create a Text Item in the Object Navigator

1. Locate the block in which you want to create the item.
2. Select the Items node.
3. Click the Create icon.

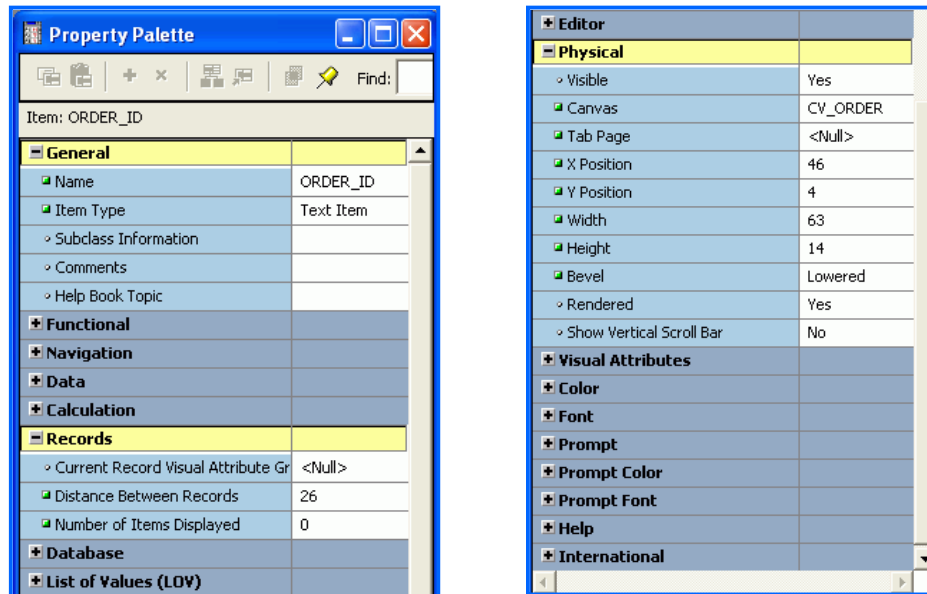
A new item entry is displayed in the Object Navigator.

4. Double-click the icon to the left of the new item entry.
The Property Palette appears.

5. Set all item properties as required, keeping the Type property set to Text Item.

Note: To display an item at run time, you must assign the item to a canvas. Do this in the Property Palette of the text item by setting the Canvas property to the desired canvas.

Modifying the Appearance of a Text Item: General and Physical Properties



ORACLE

7 - 6

Copyright © 2009, Oracle. All rights reserved.

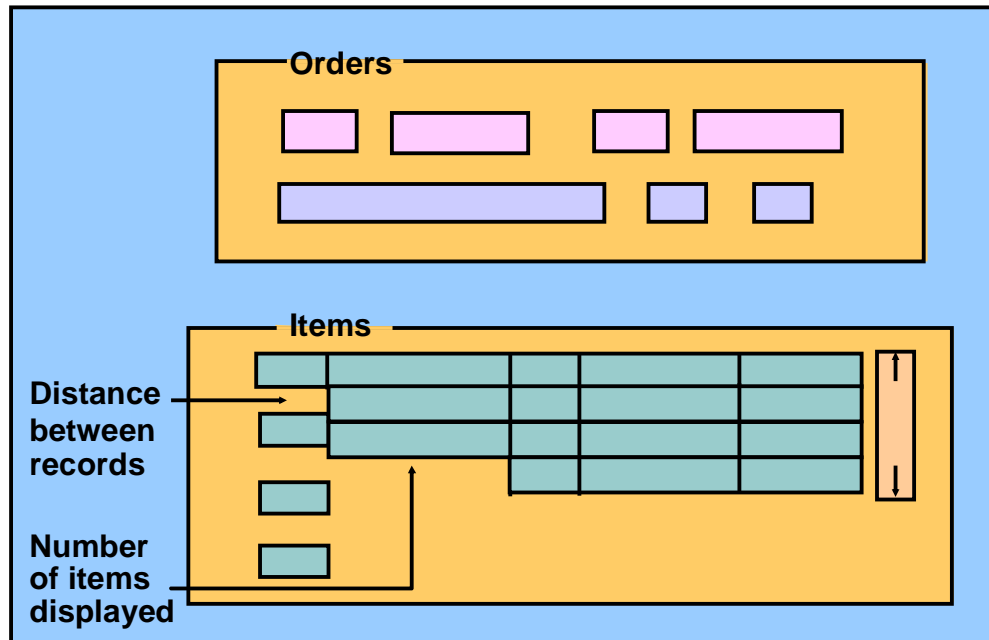
Modifying the Appearance of a Text Item: General and Physical Properties

The properties of an item are divided into several groups, as shown in the screenshots of the Property Palette in the slide.

You can affect the way the text item is displayed by altering its General, Physical, Records, Font, and Color group properties. To view descriptions of any of these properties, click the property in the Property Palette and press F1. The following are some of the properties:

- **General:**
 - **Item Type:** Selects the type of item you want to create (pop-up list)
- **Physical:**
 - **Visible:** Determines whether the item is displayed
 - **Canvas:** Determines the canvas on which the item is displayed. If left unspecified, the item is said to be a *Null canvas* item, and will not display at run time or in the Layout Editor.
 - **X and Y Position:** Sets the x and y coordinates of the item relative to the canvas
 - **Width and Height:** Sets the width and height of the item in the current form coordinate units
 - **Bevel:** Controls appearance of bevel around the item; can also be set to Plain (flat) or None

Modifying the Appearance of a Text Item: Records Properties



ORACLE

7 - 7

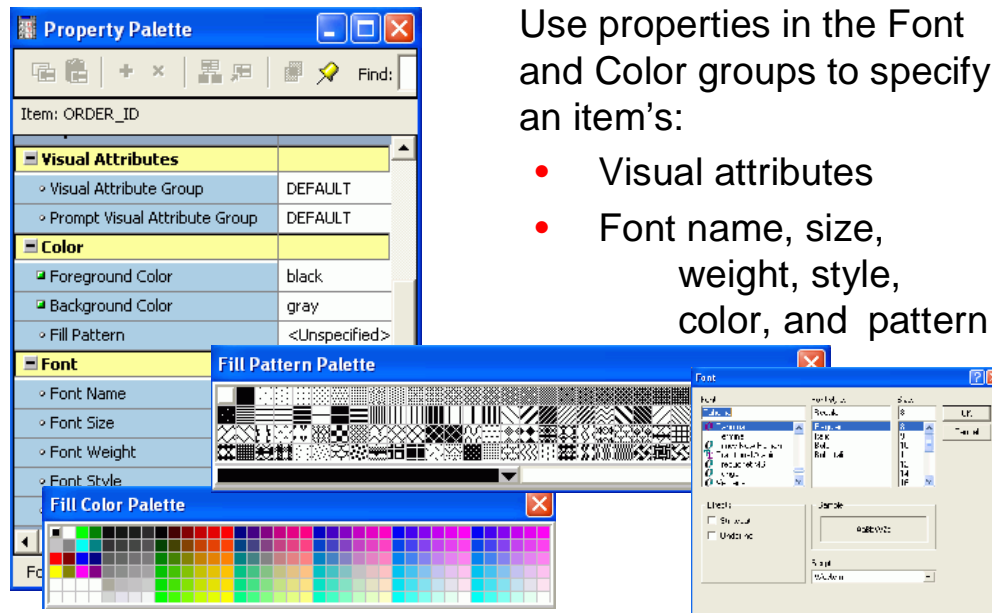
Copyright © 2009, Oracle. All rights reserved.

Modifying the Appearance of a Text Item: Records Properties

- **Records:**
 - **Current Record Visual Attribute Group:** Specifies the name of the visual attribute to use when the item is part of the current record
 - **Distance Between Records:** Specifies the amount of space between instances of the item in a multirecord block
 - **Number of Items Displayed:** Specifies the number of item instances that are displayed for the item when the item is in a multirecord block

The graphic shows four records displayed in the Items block. The first item has a nonzero value for the Distance Between Records property, showing that you can specify this setting on a per-item basis.

Modifying the Appearance of a Text Item: Font and Color Properties



Use properties in the Font and Color groups to specify an item's:

- Visual attributes
- Font name, size, weight, style, color, and pattern

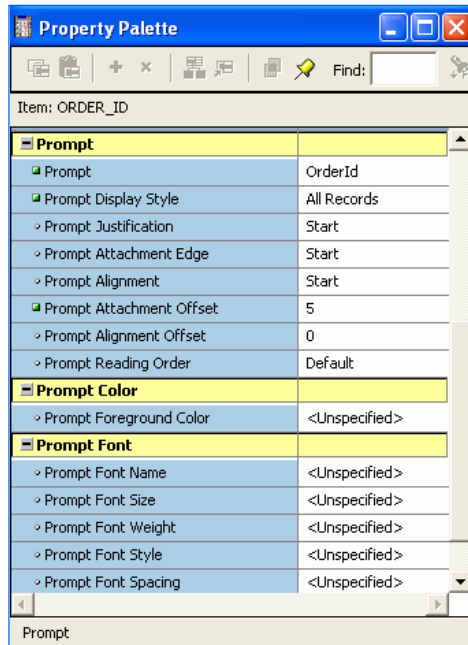
Modifying the Appearance of a Text Item: Font and Color Properties

- **Visual Attributes:** You set the Visual Attribute Group for an item or its prompt to specify how the visual attributes are derived (select DEFAULT or a named Visual Attribute).
- **Color:** The properties in this section specify the foreground color, background color, and fill pattern for the item. You select these from color and pattern pickers.
- **Font:** The properties in this section determine the font used for the item, along with its size, weight, style, and spacing. You can double-click the Font group to display a Font dialog box enabling you to set all these properties at once, or you can click the individual properties to select each from an appropriate control, such as a pop-up list or list of values (LOV).

The screenshots show the Property Palette with the fill pattern and fill color pickers and the font dialog box.

Note: When the form module does not contain any named Visual Attribute objects, the pop-up list for the Visual Attribute Group property shows only DEFAULT or unspecified. An item that has the Visual Attribute Group property set to DEFAULT, or that has individual attribute settings left unspecified, inherits those settings from the canvas to which it is assigned.

Modifying the Appearance of a Text Item: Prompts



- A prompt specifies the text label that is associated with an item.
- Several properties are available to arrange and manage prompts.
- Use prompt properties to change the appearance of an item prompt.

ORACLE

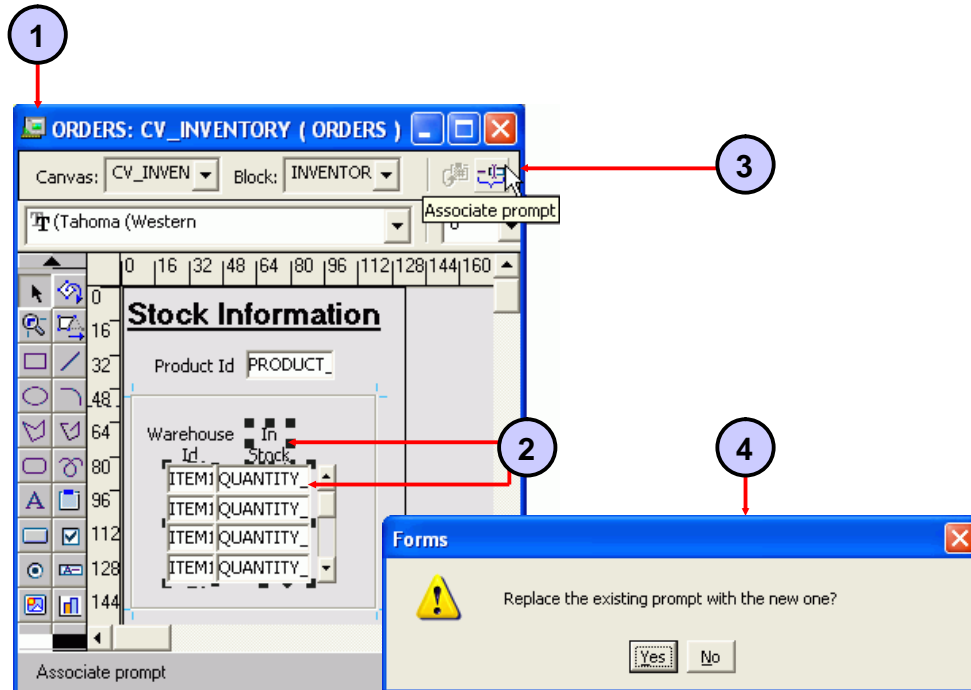
Modifying the Appearance of a Text Item: Prompts

You can control the appearance of the prompt, or label, of a text item using properties in the following groups:

- **Prompt:** You can set the prompt text and other properties, such as:
 - **Display Style:** Pop-up list with choices of First Record, Hidden, and All Records
 - **Attachment Edge:** Specifies the item edge to which the prompt is attached
 - **Attachment Offset:** Specifies the distance between the item and its prompt
- **Prompt Color:** The property in this section specifies the foreground color for the item prompt. You can select this from a color picker.
- **Prompt Font:** The properties in this section determine the font that is used for the item prompt, along with its size, weight, style, and spacing. You can double-click the Prompt Font group to display a Font dialog box, which enables you to set all these properties at once. Alternatively, you can click the individual properties to select each from an appropriate control, such as a pop-up list or LOV.

The screenshot shows the Property Palette for a text item with the Prompt groups highlighted.

Associating Text with an Item Prompt



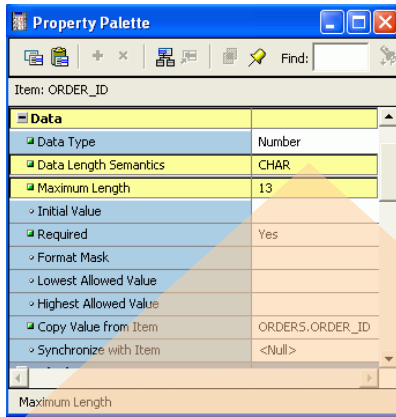
Associating Text with an Item Prompt

The Forms Builder Layout Editor has a tool called Associate Prompt, which enables you to create a prompt for an item using any boilerplate text displayed in the editor. To create a prompt-item association using the Associate Prompt tool, perform the following steps:

1. Open the Layout Editor window.
2. Select the item and boilerplate text you want as the item's prompt in the editor.
3. Click the Associate Prompt tool.
4. If you are replacing an existing prompt, click Yes in the dialog box.

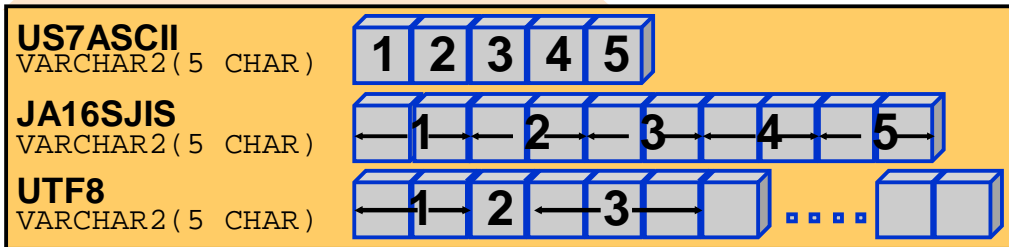
The screenshot in the slide shows the Layout Editor with a text item and boilerplate text selected. The Associate Prompt tool has been selected. A dialog box appears asking if the existing prompt should be replaced.

Controlling the Data of a Text Item



Use the properties in the Data group to control the data:

- Type
- Length
- Format
- Value



ORACLE

Controlling the Data of a Text Item

The properties in the Data group of the text item are used to control the way data is displayed and entered. You can see descriptions of any of these properties by clicking the property in the Property Palette, and then by pressing F1. The following are some of the properties:

- **Data Type:** This enables you to choose CHAR, DATE, DATETIME, and NUMBER; the others listed are for backward compatibility.
- **Data Length Semantics:** You can set to Null, BYTE, or CHAR to be compatible with multiple character sets. If Data Length Semantics is CHAR, the correct amount of storage will be automatically allocated as required for the Maximum Length with either a single-byte or multibyte character set.
- **Maximum Length:** You can use this to specify the maximum length of the data value that can be stored in an item. If the Maximum Length exceeds the display width of the item, Forms automatically enables the end user to scroll horizontally.

Note: In the example in the slide, whether the form operator is using a single-, double-, or variable-byte character set, the right amount of storage is allocated. To hold the same value if the Data Length Semantics had been set to BYTE, the Maximum Length would have needed to be 5 for single-byte, 10 for double-byte, and an unknown value for a variable-byte character set.

Controlling the Data of a Text Item: Format

Format masks:

- Standard SQL formats, such as:
 - Dates `FXDD-MON-RR`
 - Numbers `L099G990D99`
- Nonstandard formats: Use double quotation marks for embedded characters, `" ("099") "099" - "0999`.

Note: Allow for format mask's embedded characters when defining the Width property.

Quantity	Item Total
61	2,928.00
43	4,162.40
47	3,713.00
47	1,927.00

Property Palette

Item: ITEM_TOTAL

Data Type	Number
Data Length Semantics	Null
Maximum Length	30
Initial Value	
Format Mask	999G990D99
Copy Value from Item	
Synchronize with Item	<Null>

Display format and input accepted for data in display and text item

ORACLE

7 - 12

Copyright © 2009, Oracle. All rights reserved.

Controlling the Data of a Text Item: Format

- **Format Mask:** You can specify any format mask that is valid for the data type. Use the Format Mask property to specify the format in which the user sees the item value.
 - Use standard SQL formatting syntax for dates and numbers (for example, DD/MM/RR and \$99,999.99). The screenshots show that a number such as 4162.4 with a format of 999G990D99 looks like 4,162.40 at run time.
 - Enclose non-SQL-standard embedded characters within double quotation marks (for example, hyphen [-] and comma [,]).

It is recommended that you avoid creating individual masks if the general purpose masks suffice (see the lesson titled “Working in the Forms Environment”).

- **FX Format Mask:** The FX format mask in a date value ensures that the date is entered exactly as defined. Element D is for decimal and G is a group (thousands) separator. **Example:** With a date format of DD/MM/RR, valid entries are 10/12/00, 10 12 00, 10-DEC-00, or 101200. You can enter any character to represent the (/) in the value. Allow for the embedded characters of the format mask when defining the Width property. Embedded characters are used only for display purposes and are not stored. However, if the date format mask is FXDD/MM/RR, that same date must be entered as 10/12/00.

Note: For a complete list of format elements, see *Oracle Database SQL Language Reference 11g Release 1 (11.1)*, available on OTN.

Controlling the Data of a Text Item: Values

Initial values:

- Are used for every new record
- Can be overwritten
- Must be compatible with item's data type
- Use:
 - Raw value
 - System variable
 - Global variable
 - Form parameter
 - Form item
 - Sequence

ORACLE

7 - 13

Copyright © 2009, Oracle. All rights reserved.

Controlling the Data of a Text Item: Values

- **Required:** This specifies whether Forms will allow the item to have a null value. When you create a data block, Forms derives this value from the existence of a NOT NULL constraint on the database column, but you can change the value.
- **Lowest/Highest Allowed Value:** You can use this to specify the range of accepted values.
- **Initial Value:** This specifies the default value assigned to an item whenever a record is created; can be set to select from a sequence; must be compatible with the item data type. If the Lowest/Highest Allowed values are specified, the initial value cannot be outside the range.

Controlling the Data of a Text Item: Values (continued)

Creating an Initial Value

You can use any one of the following values to issue an initial item value whenever a new record is created:

- Raw value
Example: 340, RICHMOND
- System variable
 - Variables giving current application server *operating system* date/time:

Variable	Format
\$\$DATE\$\$	DD-MON-RR
\$\$DATETIME\$\$	DD-MON-YYYY hh:mi[:ss]
\$\$TIME\$\$	hh:mi[:ss]

- Variables giving current *database* date/time:

Variable	Format
\$\$DBDATE\$\$	DD-MON-YYYY
\$\$DBDATETIME\$\$	DD-MON-YYYY hh:mi[:ss]
\$\$DBTIME\$\$	hh:mi[:ss]

- Global variable
Example: :GLOBAL.CUSTOMER_ID
- Form parameter
Example: :PARAMETER.SALES_REP_ID
- Form item
Example: :ORDERS.ORDER_ID
- Sequence
The initial value can reference a sequence in the database. Forms automatically writes generated sequence numbers into the text item. To use a sequence, enter:
:SEQUENCE.<sequence name>.NEXTVAL
Example: :SEQUENCE.ORDERS_SEQ.NEXTVAL

Controlling the Data of a Text Item: Copy Value from Item

The screenshot shows the Oracle Forms Builder interface. On the left, the 'Property Palette' is open for the item '<data_block_name>.<item_name>'. The 'Copy Value from Item' property is highlighted, and its value is set to 'DEPARTMENTS.DEPARTMENT_ID'. Below the palette, a table titled 'Employee' is displayed. The table has columns: ID, Last Name, First Name, Title, and Dept ID. The data rows are:

ID	Last Name	First Name	Title	Dept ID
3	Nagayama	Midori	VP, Sales	31
11	Magee	Colin	Sales Rep	31

On the right, a detail block for the 'Employee' block is shown. It contains a 'Dept' section with 'ID' (value 31) and 'Region ID' (value 1), and a 'Name' section with the value 'Sales'. An arrow points from the 'Dept ID' value in the detail block to the 'Copy Value from Item' property in the Property Palette, indicating the source of the data.

ORACLE

7 - 15

Copyright © 2009, Oracle. All rights reserved.

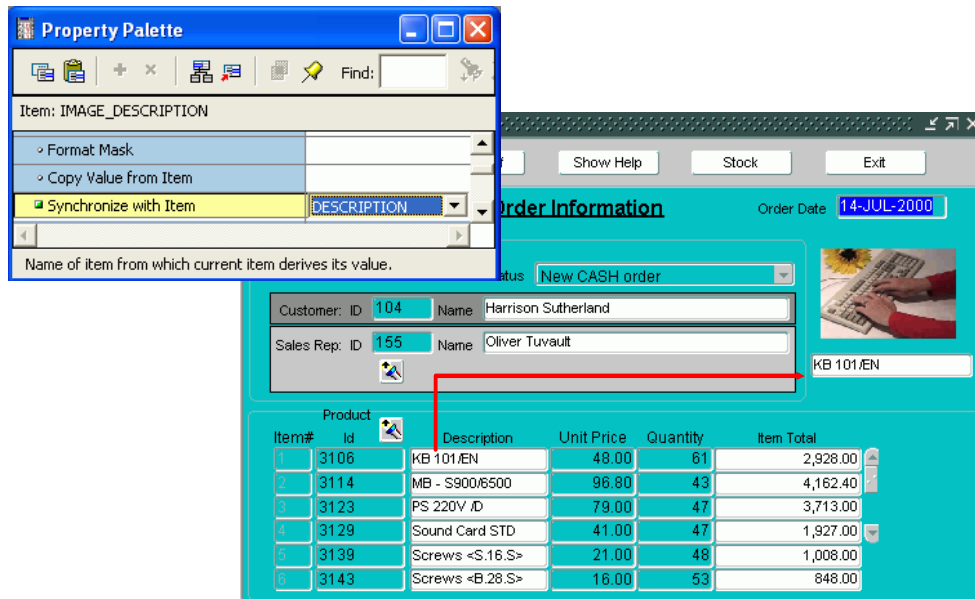
Controlling the Data of a Text Item: Copy Value from Item

- **Copy Value from Item:** This specifies the source of the value that Forms uses to populate the item. When you define a master-detail relation, Forms Builder sets this property automatically on the foreign key items in the detail block. In such cases, the “Copy Value from Item” property names the primary key item in the master block whose value gets copied to the foreign key item in the detail block whenever a detail record is created or queried.

Note: The text item should disable input; otherwise, the user can violate the foreign key relationship. To prevent this, set the Enabled property to No for the foreign key item, or do not display it at all.

The example in the slide shows that the department ID item on the Employee block (detail) has “Copy Value from Item” set to DEPARTMENTS . DEPARTMENT_ID from the master block.

Controlling the Data of a Text Item: Synchronize with Item

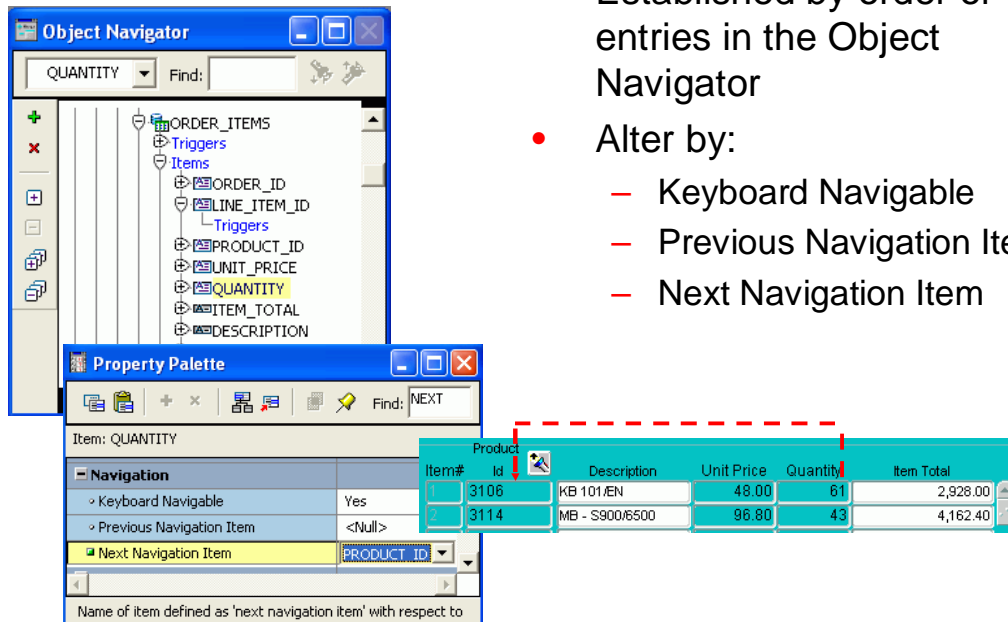


Controlling the Data of a Text Item: Synchronize with Item

- **Synchronize with Item:** You can use this property to specify the name of the item from which the current item should derive its value and to synchronize the values of the two items, so that they effectively mirror each other. When the end user or the application changes the value of either item, the value of the other item also changes.

The screenshots show that the caption of the product picture, which is on the control block, is synchronized with the product description in the data block.

Altering Navigational Behavior of Text Items



The screenshot shows the Oracle Forms Object Navigator and Property Palette. The Object Navigator displays a tree structure for the 'QUANTITY' item, including 'ORDER_ITEMS', 'Triggers', 'Items', 'ORDER_ID', 'LINE_ITEM_ID', 'Triggers', 'PRODUCT_ID', 'UNIT_PRICE', 'QUANTITY', 'ITEM_TOTAL', and 'DESCRIPTION'. The Property Palette shows the 'Navigation' group for the 'QUANTITY' item. The 'Next Navigation Item' property is set to 'PRODUCT_ID'. A red dashed line indicates the navigation path from 'QUANTITY' to 'PRODUCT_ID' in the data table.

Item#	Id	Description	Unit Price	Quantity	Item Total
3106		KB 101/EN	48.00	61	2,928.00
3114		MB - S900/6500	96.80	43	4,162.40

- Established by order of entries in the Object Navigator
- Alter by:
 - Keyboard Navigable
 - Previous Navigation Item
 - Next Navigation Item

Altering the Navigational Behavior of Text Items

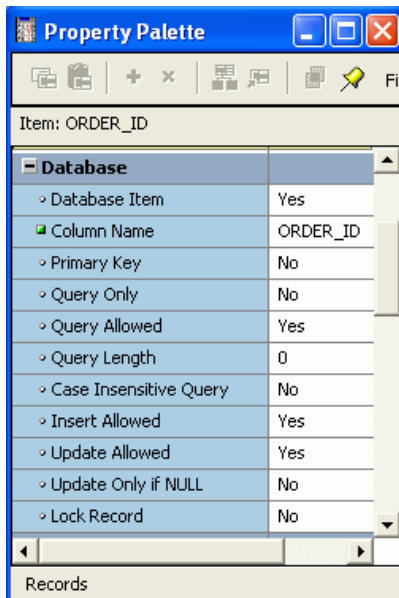
You can see the default navigational sequence of items in the Object Navigator because the item entries are displayed in the navigational order. However, you can also use the Navigation group properties to control the navigational behavior of a text item.

Navigation Property	Function
Keyboard Navigable	Determines whether you can navigate to an item during default navigation with the function keys or menu items and place focus on it. When this property is set to No, Forms skips over the item and enters the next navigable item in the default navigation sequence.
Previous Navigation Item	Determines the previous item to be visited when you navigate out of the current item
Next Navigation Item	Determines the next item to be visited when you navigate out of the current item

The slide graphic shows setting Next Navigation Item for the Quantity item to navigate to the Product_Id item.

Note: The next or previous navigation item must be in the same block as the current item.

Enhancing the Relationship Between Text Item and Database



Use the properties in the Database group to control:

- Item's data source—base table item or control item
- Query, insert, and update operations on an item
- Maximum query length
- Query case

ORACLE

Enhancing the Relationship Between Text Item and Database

You can alter or enhance the way in which a text item interacts with its corresponding database column by setting the Database group properties. Some of these are:

- **Database Item:** Indicates whether the item is a database column
- **Query/Insert/Update Allowed:** Controls whether data manipulation language (DML) operations are allowed
- **Query Length:** Specifies the maximum length of query criterion in Enter Query mode
- **Case Insensitive Query:** Controls whether case is recognized in query processing

The screenshot shows the Database group on the Property Palette for a text item.

Note: When you create an item in a data block, Forms Builder assumes that the item is a data item, sets its Database Item property to Yes, and automatically includes it in any `SELECT`, `UPDATE`, and `INSERT` statements issued to the database. If an item that you are creating is a control item, you must explicitly set its Database Item property to No.

Adding Functionality to a Text Item

The screenshot shows the 'Property Palette' for a text item named 'ORDER_ID'. The 'Functional' group is expanded, showing properties like Enabled, Justification, Implementation Class, Multi-Line, Wrap Style, Case Restriction, Conceal Data, Keep Cursor Position, Automatic Skip, and Popup Menu. The 'Enabled' property is set to 'Yes', 'Justification' is 'Start', 'Case Restriction' is 'Mixed', and 'Wrap Style' is 'Word'. The 'Item' property is set to 'ORDER_ID'.

Below the palette, a diagram illustrates the form layout. A blue box labeled 'Case Restriction=Upper' contains a yellow box labeled 'Order ID' with the value '100'. A label 'Enabled=No' points to the 'Order ID' box. To the right, a yellow box labeled 'Payment Type' contains the value 'CREDIT'. Below this, an orange box labeled 'Item' contains a table with the following data:

ID	Product ID	Price	Quantity	Item Total
1	10011	135	500	67,500.00
2	10013	380	400	152,000.00

A label 'Justification = Right' points to the 'Item Total' column of the table. A label 'Justification = Start' points to the 'Order ID' box.

Adding Functionality to a Text Item

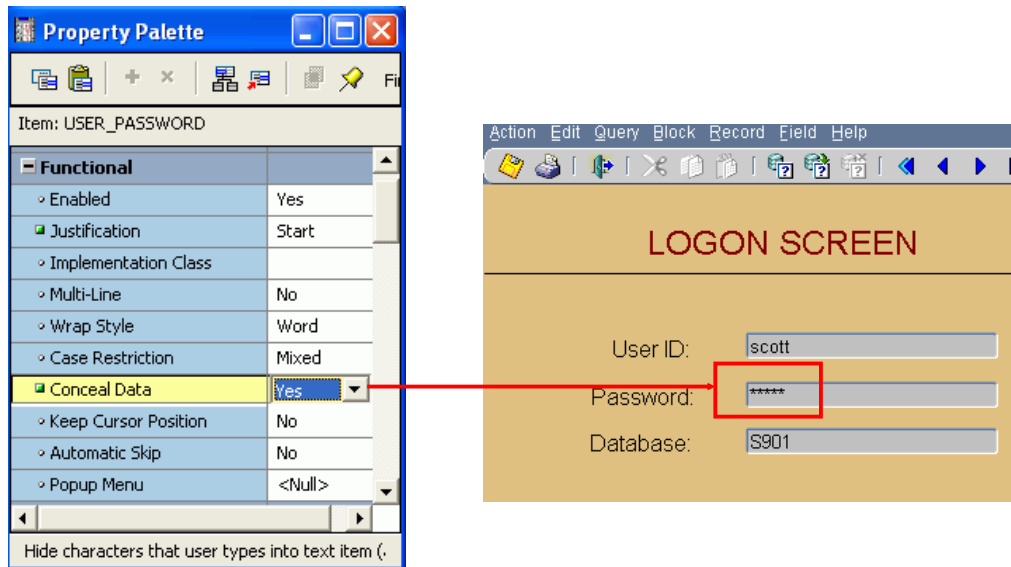
Augment the default functionality of a text item by introducing some of the additional features you can set in the Functional group of the Property Palette. Some of these are depicted in the slide or will be discussed in the subsequent slides. For descriptions of other properties in the Functional group, select the property in the Property Palette and press F1.

The screenshot shows the Functional group in the Property Palette for a text item. The graphics illustrate how setting the Enabled, Case Restriction, and Justification properties affect the functionality of the text item at run time.

Note

- If the Enabled property set to No, the item is disabled. If you want the item to appear normally but do not want users to change it, do the following:
 - Set Insert Allowed to No.
 - Set Update Allowed to No.
 - Set Enabled to Yes.
- A pop-up menu is a context-sensitive menu that enables users to access common functions and commands quickly. It is a top-level object in the Object Navigator and belongs to a form module (as opposed to a form menu, which belongs to a separate menu module).

Adding Functionality to a Text Item: Conceal Data Property



ORACLE

7 - 20

Copyright © 2009, Oracle. All rights reserved.

Adding Functionality to a Text Item: Conceal Data Property

Conceal Data: Hides characters that the operator types into the text item. This setting is typically used for password protection. Select Yes to disable the echoing back of data entered by the operator; with this setting, the entered value is displayed as an asterisk for each character entered as shown in the slide.

Note: Conceal Data set to Yes is valid only for single-line text items.

Adding Functionality to a Text Item: Keyboard Navigable and Enabled

The Keyboard Navigable and the Enabled properties work in concert with one another:

- Set both properties to allow or disallow navigation and interaction with text item.
- When Enabled is set to Yes, Keyboard Navigable can be set to Yes or No.
- When Enabled is set to No, the item is always nonnavigable.

ORACLE

7 - 21

Copyright © 2009, Oracle. All rights reserved.

Adding Functionality to a Text Item: Keyboard Navigable and Enabled

You can set the Keyboard Navigable and Enabled properties for items to specify whether operators can navigate to and interact with them. The Enabled property determines whether end users can use the mouse to manipulate an item. The following table describes the behavior of combinations of these settings:

Enabled	Keyboard Navigable	Navigation Behavior
Yes	Yes	Item is included during default navigation. The item can be navigated to and manipulated with the mouse.
Yes	No	Item is excluded during default navigation. The item can be navigated to and manipulated with the mouse.
No	No	Item is excluded during default navigation. The item cannot be navigated to and manipulated with the mouse.
No	Yes	Item is excluded during default navigation. The item cannot be navigated to and manipulated with the mouse. The Keyboard Navigable property is also effectively set to No.

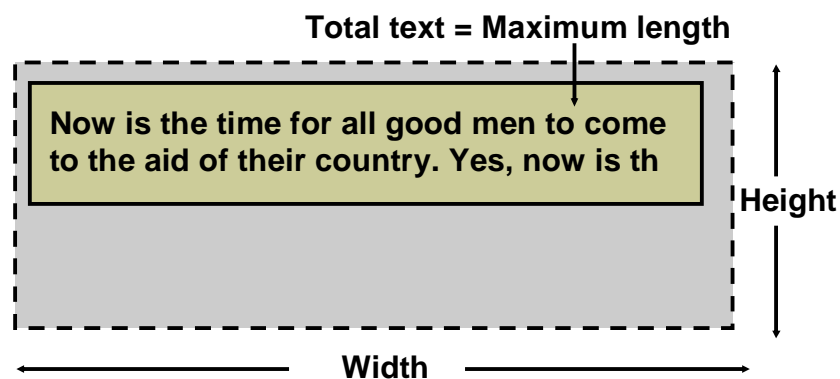
Adding Functionality to a Text Item: Multi-Line Text Items

Multi-Line:

- Possible values: Yes/No
- Cannot use for numeric or date values
- Should also set width, height, font size, maximum length

Wrap Style possible values:

- None
- Character
- Word



ORACLE

Adding Functionality to a Text Item: Multi-Line Text Items

Multi-Line: Determines whether the text item displays in a single-line or multi-line region. Use multi-line text items to display and/or edit such items as addresses, comments, or descriptions. The data in a multi-line text item must be of Char, Alpha, or Long data type; not numeric or date.

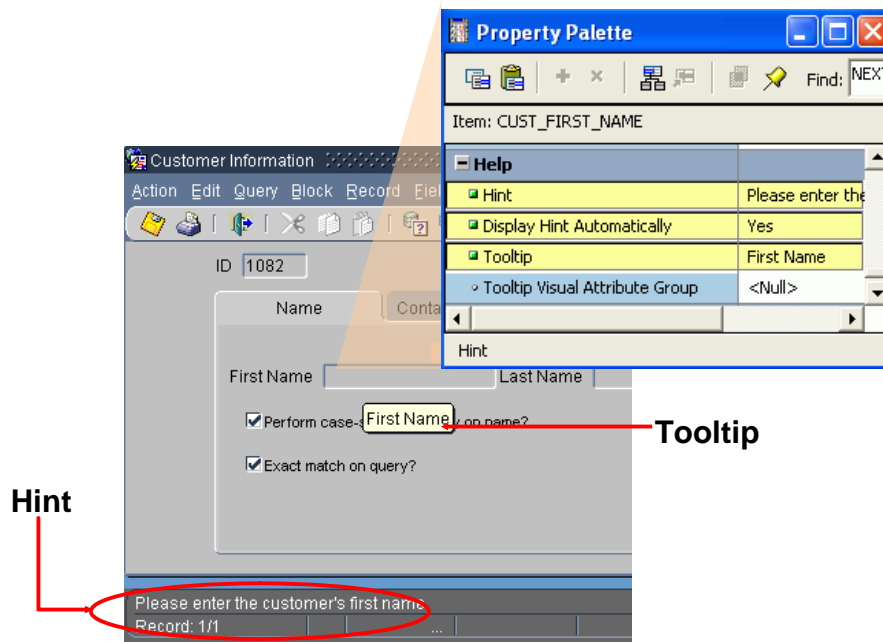
Setting the Multi-Line property to Yes enables a text item to store multiple lines of text, but it does not automatically make the item large enough to display multiple lines. You can set the Width, Height, Font Size, and Maximum Length properties to ensure that the desired number of lines and characters are displayed.

The graphic in the slide illustrates that even if the height and width of the text item enable you to enter a lot of text, the maximum length limits the length of the value that can be stored in the text item. When a value that is too long for the text item is retrieved from the database, truncation occurs.

Wrap Style: For multi-line text items, specifies how text is displayed when a line of text exceeds the width of a text item or editor window

Note: Setting right or center justification for scrollable text items may result in values being hidden from the user.

Displaying Helpful Messages: Help Properties



Displaying Helpful Messages: Help Properties

You can use the Help group properties to provide context-sensitive help to users:

Help Property	Function
Hint	Writes item-specific Help text that is displayed on the message line at run time. The Help text is available when input focus is on the item.
Display Hint Automatically	Determines whether the hint for the item is displayed automatically. If set to No, the hint displays only when the operator clicks Help or selects the Help command on the default menu.
Tooltip	Help text that should appear in a small box beneath the item when the cursor moves over the item. The item does not need to have input focus for the tooltip to appear.
Tooltip Visual Attribute Group	Specifies Visual Attribute to use for the tooltip

The screenshots show the hint appearing at the bottom of the form when a user navigates to its text item, while the tooltip appears in the vicinity of its text item when the cursor moves over the text item at run time.

Summary

In this lesson, you should have learned that:

- Text items are interface objects that usually correspond to database columns
- You can create a text item with:
 - The Text Item tool in the Layout Editor
 - The Create icon in the Object Navigator
 - The Data Block Wizard

ORACLE

7 - 24

Copyright © 2009, Oracle. All rights reserved.

Summary

This lesson showed you how to create and modify a text item that Forms Builder creates for each column flagged for inclusion in a data block.

Summary

- You can modify a text item in its Property Palette:
 - General, Records, and Physical properties control the appearance of the text item
 - Data properties control the length, data type, format, and other aspects of the data
 - Navigation properties control how to navigate to and from a text item
 - Database properties specify the relationship between the text item and its corresponding database column
 - Functional properties control how the text item functions
 - Help properties specify the display of helpful messages

ORACLE

7 - 25

Copyright © 2009, Oracle. All rights reserved.

Summary (continued)

In particular, text items have properties that enable you to do the following:

- Modify their appearance.
- Control the data stored in the item.
- Alter navigational behavior.
- Enhance the relationship with the database.
- Add functionality.
- Include Help information.

Practice 7: Overview

This practice covers the following topics:

- Deleting text items
- Modifying text item properties
- Creating text items

ORACLE

7 - 26

Copyright © 2009, Oracle. All rights reserved.

Practice 7: Overview

In this practice session, you will create text items, alter the behavior and the appearance of text items, and delete text items.

- Delete the Region_Id item in the Customers form.
- Using the Property Palette, change the properties of several text items in the CUSTOMERS data block to change their run-time appearance. Save and run the form after the changes are applied.
- In the Orders form, create new text items to hold the customer name and sales rep name values in the ORDERS block, and set the suggested properties. Change additional text item properties in the ORDERS, ORDER_ITEMS, and INVENTORIES data blocks to change their run-time appearance and behavior. Save and run the form after the changes are applied.

8

Creating LOVs and Editors

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe lists of values (LOVs) and editors
- Design, create, and associate LOVs with text items in a form module
- Create editors and associate them with text items in a form module

ORACLE

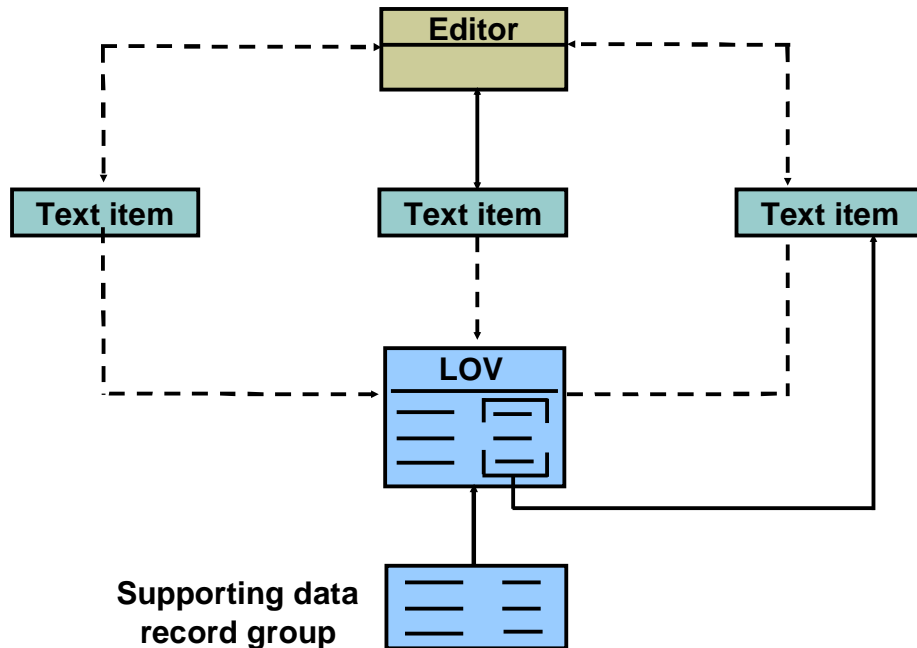
8 - 2

Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

With Oracle Forms Builder, you can enhance your application with lists of available values and text editors to supplement the text item object. In this lesson, you learn how to create LOVs and text editors, and to associate them with items in your application.

Overview of LOVs and Editors



ORACLE

8 - 3

Copyright © 2009, Oracle. All rights reserved.

Overview of LOV and Editors

LOVs and editors are objects in a form module that you can associate with text items to enhance input. Each opens its own window when activated at run time.

LOVs enable users to choose a value from a static or dynamic list, while editors provide a larger area for text entry with search and replace capability. Both are defined at the form level, which means that you can use them to support text items in any block of the form module.

The slide graphics depict three text items on a form. The first and second items use the same editor, while the second and third items use the same LOV. The LOV takes its values from a supporting record group. In this lesson, you learn how to create these objects and attach them to text items.

What Are LOVs?

- List of values for text items
- Dynamic or static list
- Independent of single text items
- Flexible and efficient

ORACLE

8 - 4

Copyright © 2009, Oracle. All rights reserved.

What Are LOVs?

An LOV is a scrollable pop-up window that enables a user to pick the value of an item from a multicolumn dynamic list. The user can reduce the lines displayed in the list by simple automatic reduction techniques, or by search strings.

Each line in an LOV can present several field values, with column headings above. You can design your LOV to retrieve some or all of the field values from the line chosen by the user and place them into form items.

LOVs have the following qualities:

- **Dynamic:** The list entries can change to reflect changes in the source data.
- **Independent:** The designer can invoke an LOV from any text item, or from outside a text item if called programmatically.
- **Flexible:** You can use the same LOV to support several items, if appropriate (for example, product_ID, product_name).
- **Efficient:** You can design LOVs to reuse data already loaded into the form, instead of accessing the database for every call. This is useful where data is relatively static.

Using an LOV at Run Time

At run time:

- Status bar message (lamp)
- List to select value for data entry
- Automatic reduction
- Search



LOV lamp

ORACLE

8 - 5

Copyright © 2009, Oracle. All rights reserved.

Using an LOV at Run Time

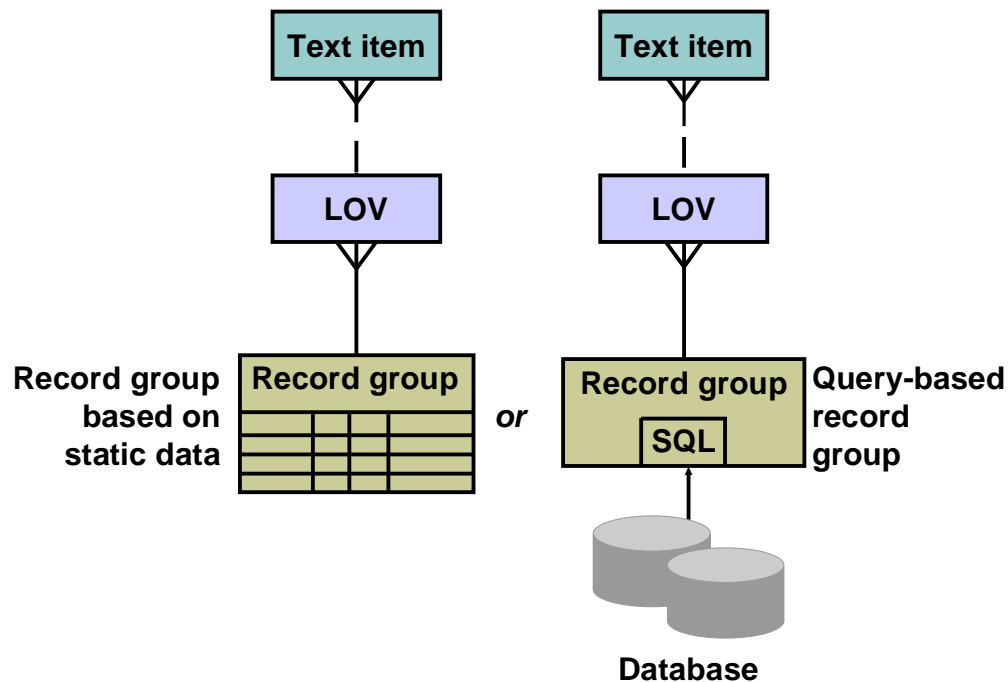
When a text item has an LOV attached, the List of Values lamp displays on the status line while the cursor is on the item, as shown in the screenshot.

To use the LOV:

1. Either click List of Values, or select Edit > Display List to invoke the LOV.
2. Select an entry in the displayed list. You can enter characters to automatically reduce the list, or enter a search string in the Find field.
3. Click OK to retrieve the line value.

Note: Automatic reduction works by comparing the search string entered with the values displayed in the first column of the LOV. If you start your search criteria with a % symbol, Forms performs a search on all LOV columns.

LOV-Related Objects



ORACLE

8 - 6

Copyright © 2009, Oracle. All rights reserved.

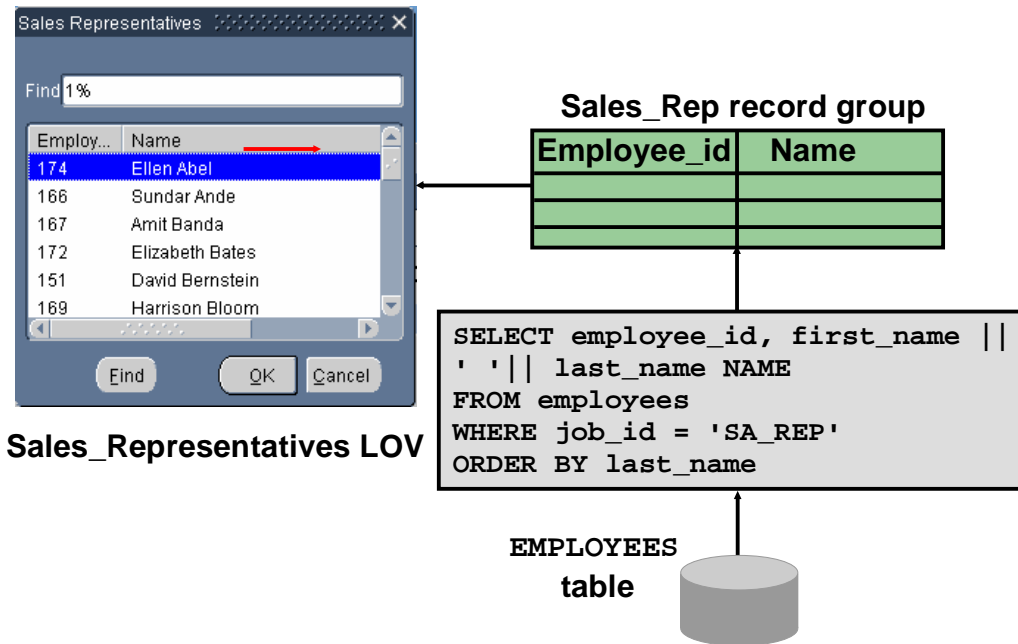
LOV-Related Objects

When you build an LOV, consider the following objects:

- **Record group:** A Forms Builder object that is used to store the array of values that are presented by an LOV (The record group can be created first or as part of the LOV creation process if based on a query.)
- **LOV:** The list itself, which presents one or more column values from the supporting record group in the LOV window (It enables the user to select values, and then write values back to specified items or variables.)
- **Text items:** The main text item that you attach to an LOV is usually one that the LOV returns a value to. You can call the LOV from this item to provide possible values for it. A single LOV can return values to several items. You can attach the LOV to any text item from which the same list of values needs to be viewed, whether or not it will receive a value.

The slide shows two types of LOVs. The left side of the slide shows a text item with an attached LOV that is based on a static record group. The right side of the slide shows a text item with an attached LOV that uses a SQL query to define the record group.

LOVs and Record Groups



LOVs and Record Groups

A record group is a column-and-row structure stored within Forms Runtime memory and is similar to the structure of a database table. It holds records that can be reused by other text items, therefore reducing repeated access to external data.

Record groups can be designed to contain static values. Alternatively, they can be populated programmatically at run time or, most commonly, populated by a SQL query. In this lesson, you use record groups to support LOVs.

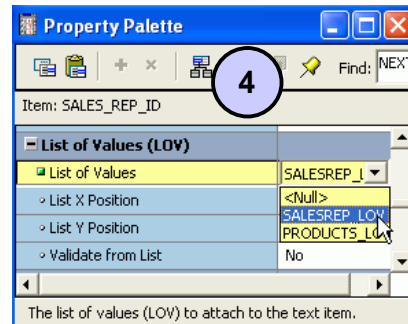
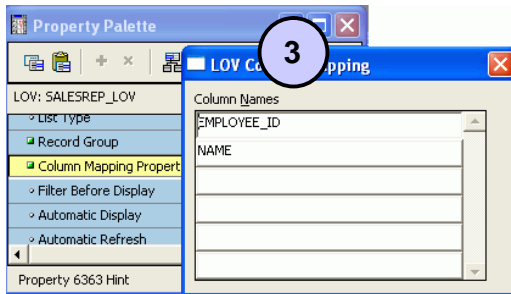
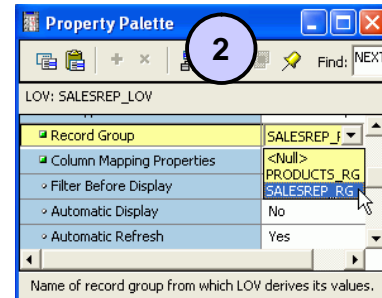
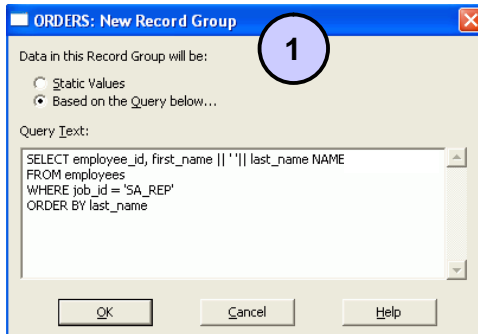
Record groups can provide the following:

- Data that is presented by LOVs
- Data for dynamic list items

Note: Because LOVs and record groups are separate objects, you can create multiple LOVs based on the same record group.

The screenshot shows an LOV for sales representatives that displays employee ID and name. The graphics in the slide show that the data for this LOV comes from a record group that is based on a SQL query, so is therefore dynamically retrieved from the database.

Creating an LOV Manually



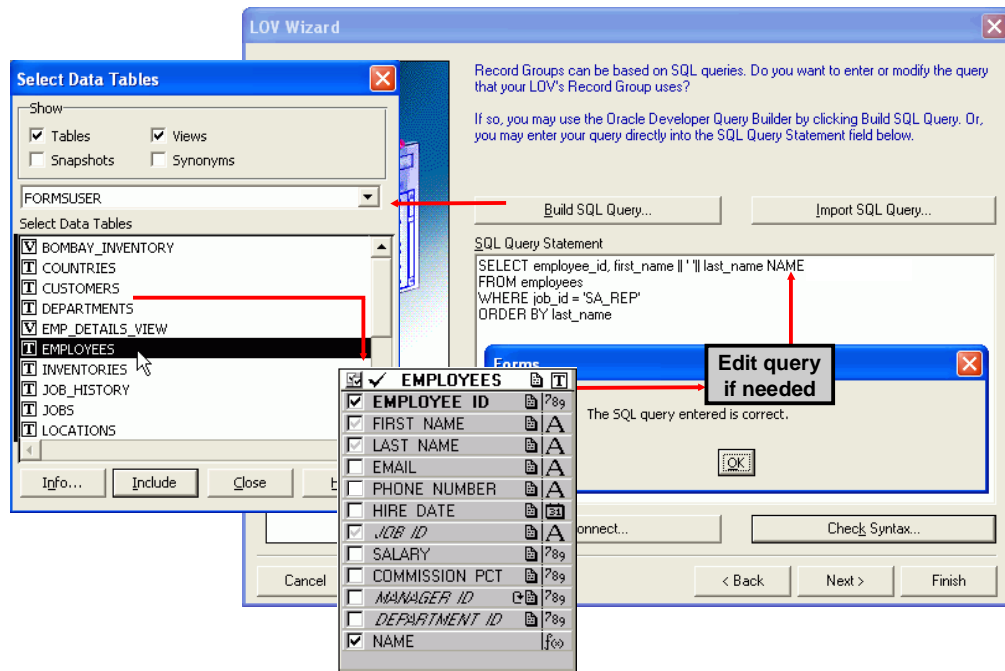
Creating an LOV Manually

Because Forms Builder has an LOV Wizard for you to use in creating LOVs and their associated record groups, you may never need to create an LOV manually. However, knowing how to do so helps you understand how to set the properties of the record group, the LOV, and the item to which it is attached, even if using the wizard.

The steps to create an LOV manually are the following:

1. Create the record group. You will need to enter the query on which the record group is based.
2. Create the LOV and set its Record Group property to the appropriate record group.
3. Set the LOV Column Mapping Property. You must enter the columns and their headings, and then select a return item for each item that you want to populate from the LOV.
4. Assign the LOV to any text items from which you want the LOV to be available.

Creating an LOV with the LOV Wizard: SQL Query Page

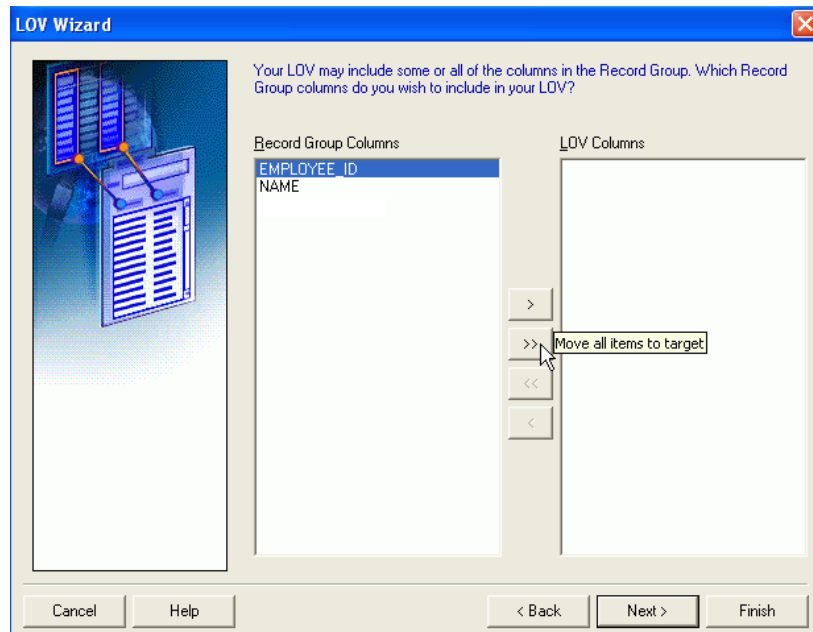


Creating an LOV with the LOV Wizard: SQL Query Page

It is easy to make a mistake or to forget one of the manual steps. This can be avoided by using the LOV Wizard, which guides you through the process. To create an LOV with the wizard, perform the following steps:

1. Launch the LOV Wizard.
The Welcome page appears. Click Next. The LOV Source page appears.
2. Specify the LOV source on the LOV Source page. Choose an existing record group or create a new one based on a query. The default option is New Record Group based on a query. Click Next to select the default. The SQL Query page appears.
3. On the SQL Query page, specify the query that is used for the record group. You cannot include a column of a complex object data type. Use one of the following three options for constructing the query:
 - Click Build SQL Query to use Query Builder, as shown in the slide. The EMPLOYEES table is used and the columns that are selected are the EMPLOYEE_ID column and a pseudocolumn called NAME, consisting of FIRST_NAME concatenated with LAST_NAME.
 - Click Import SQL Query to import the query from a file.
 - Enter the SQL syntax in the SQL Query Statement field to enter the query directly. Then click Check Syntax; a message about the validity of the query is displayed, as shown in the slide.

Creating an LOV with the LOV Wizard: Column Selection Page

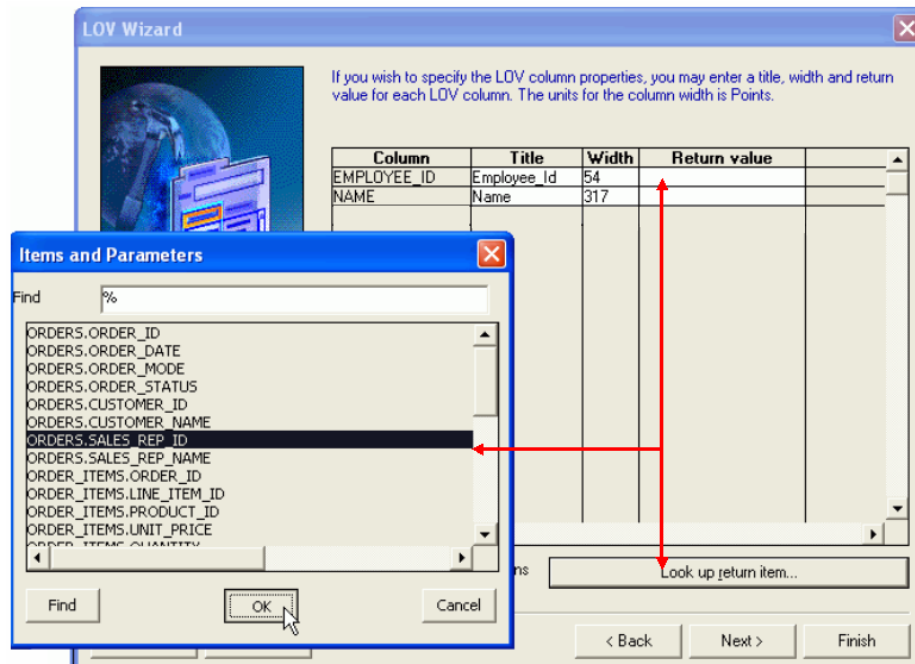


Creating an LOV with the LOV Wizard: Column Selection Page

4. On the Column Selection page, select the record group columns that you want to include in the LOV and shuttle them to the LOV Columns list, as shown in the slide. Click Next. The Column Properties page appears.

The slide shows that the wizard presents two record group columns that are available for inclusion in the LOV: `EMPLOYEE_ID` and `NAME`.

Creating an LOV with the LOV Wizard: Column Properties Page



ORACLE

8 - 11

Copyright © 2009, Oracle. All rights reserved.

Creating an LOV with the LOV Wizard: Column Properties Page

5. On the Column Properties page, specify the Title, Width, and Return value for each LOV column. Note that Return value is optional and there is a button that invokes a separate window from which you can choose an item. Click Next. The LOV Display page is displayed.

The screenshots show how to set a Return value for Employee_Id. The “Look up return item” button was clicked to invoke a separate window, and the `ORDERS.SALES_REP_ID` item is selected. This means that, when the user selects an employee from the LOV, that employee’s ID populates the Sales_Rep_Id text item in the ORDERS block.

Creating an LOV with the LOV Wizard: LOV Display Page

LOV Wizard

What title would you like to display in your LOV window?

Title: Sales Representatives

What size would you like your LOV to be? The units for the LOV size and position are Points.

Width: 180 Height: 135

Do you want Forms Runtime to position your LOV?

☒ Yes, let Forms position my LOV automatically

☐ No, I want to position it manually

Left: 0 Top: 0

Cancel Help < Back Next > Finish

ORACLE

8 - 12

Copyright © 2009, Oracle. All rights reserved.

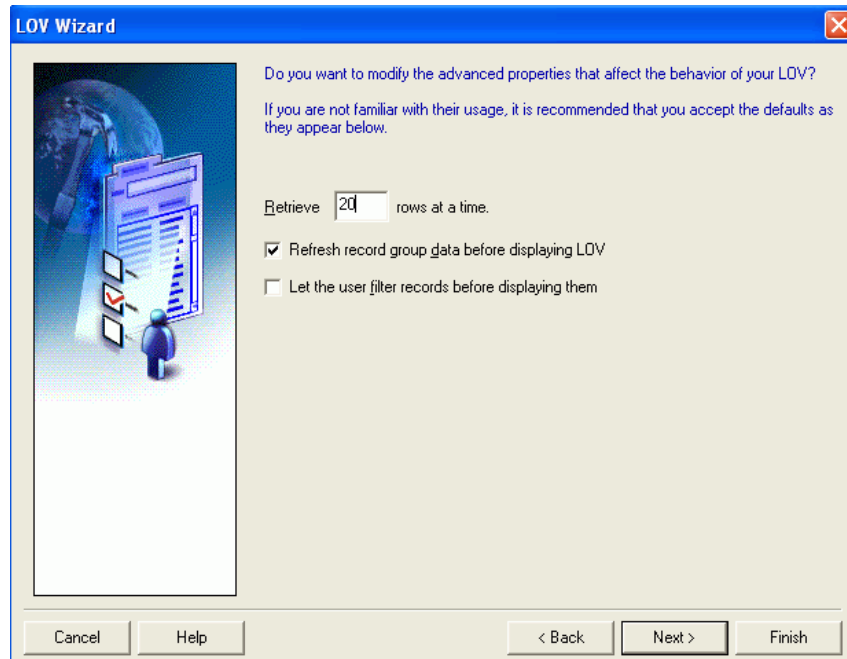
Creating an LOV with the LOV Wizard: LOV Display Page

6. On the LOV Display page, specify the title, the width, and the height of the LOV window. You can choose to display it at a set position that you manually define, or let Forms position it automatically. Click Next. The Advanced Options page appears.

In the slide, you see the screenshot of the LOV Display page of the wizard with the following selections:

- **Title:** Sales Representatives
- **Width:** 180
- **Height:** 135
- Automatic positioning option selected

Creating an LOV with the LOV Wizard: Advanced Properties Page



ORACLE

8 - 13

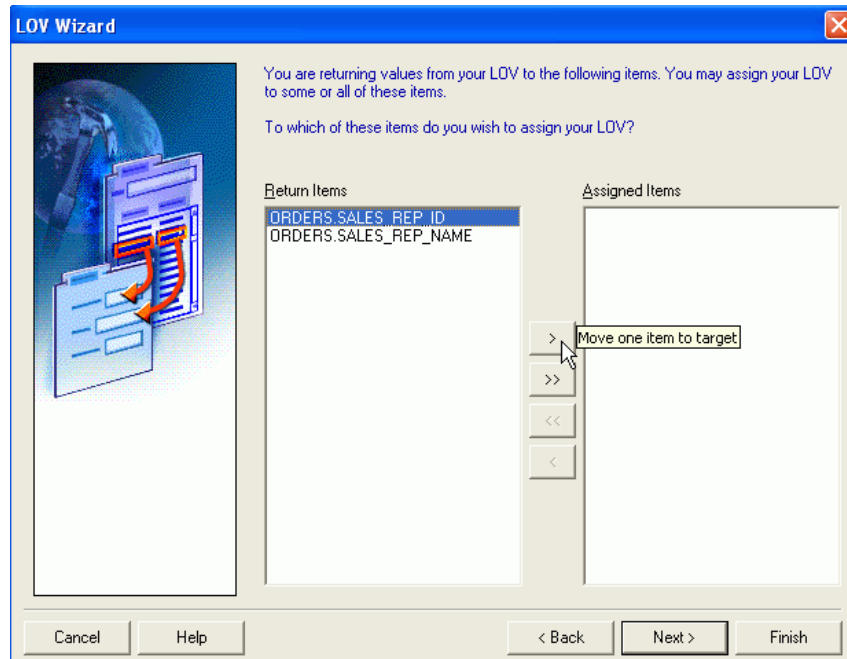
Copyright © 2009, Oracle. All rights reserved.

Creating an LOV with the LOV Wizard: Advanced Properties Page

7. On the Advanced Properties page, set the additional advanced properties for the LOV. Specify:
 - The number of records at a time to be fetched from the database; the example in the slide shows 20
 - If the LOV records should be queried each time the LOV is invoked (this check box is selected in the slide example)
 - If the user should be presented with a dialog box to add criteria before the LOV is displayed (this check box is not selected in the slide example)

Click Next. The “Assign to Item” page appears.

Creating an LOV with the LOV Wizard: “Assign to Item” Page



ORACLE

8 - 14

Copyright © 2009, Oracle. All rights reserved.

Creating an LOV with the LOV Wizard: “Assign to Item” Page

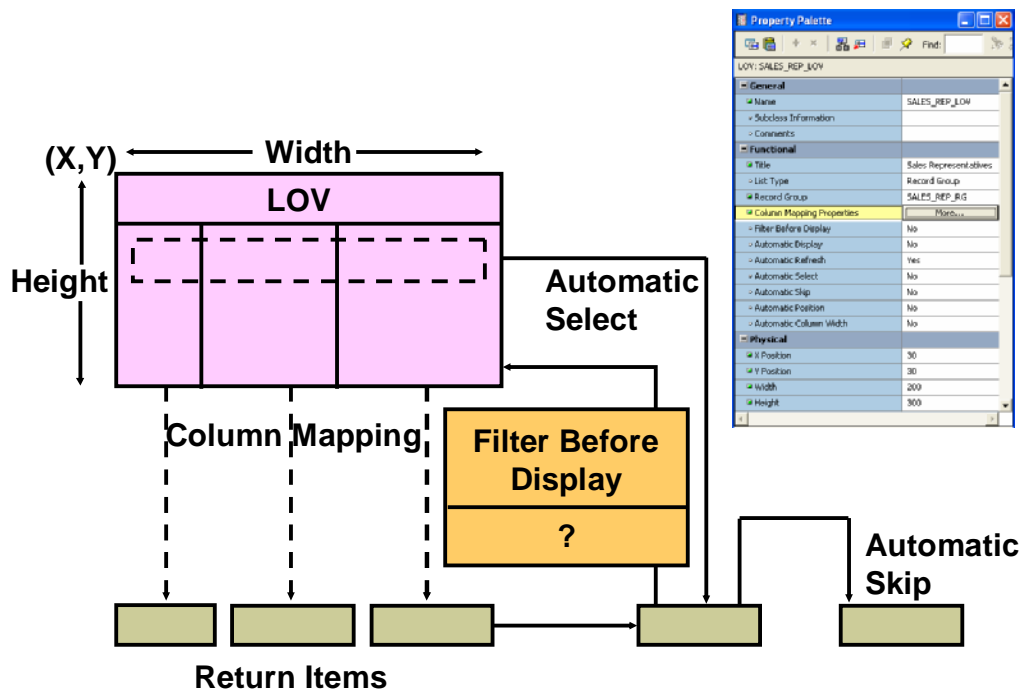
- On the “Assign to Item” page, select the items to which your LOV should be attached; the screenshot shows `ORDERS.SALES_REP_ID` selected in the available Return Items and that the user is about to shuttle that selection to the Assigned Items column. At run time, the LOV is available from the `Sales_Rep_Id` item so that operators may use it while input focus is in that item.

Click Next. The Finish page appears.

- On the Finish page, click Finish to complete the LOV creation process.

Note: The LOV Wizard is reentrant, so you can use it to modify the LOV after it is created. In the Object Navigator, select the LOV to be modified, and then select Tools > LOV Wizard from the menu.

Setting LOV Properties

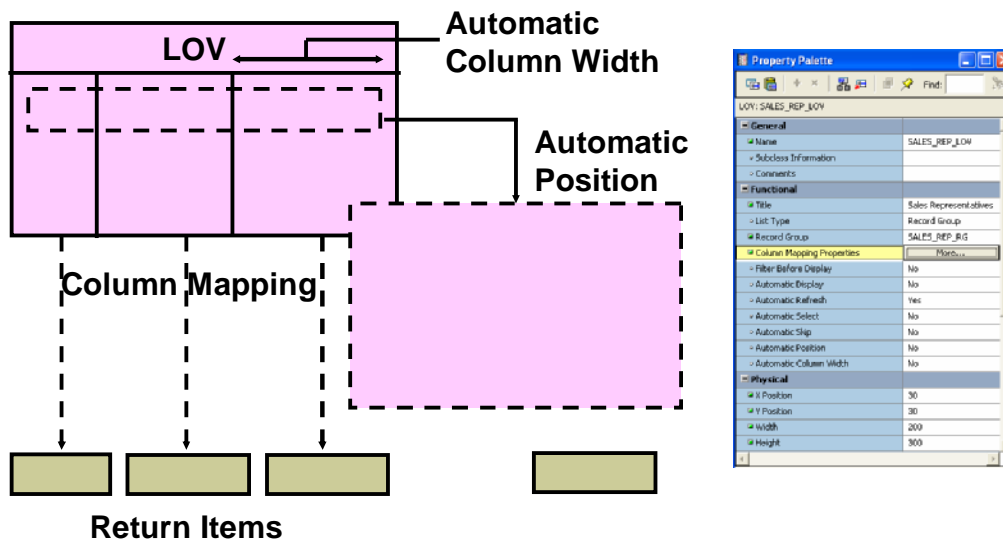


Setting LOV Properties

After you create an LOV, you can open its Property Palette to modify or further define its properties. Some of these properties are the following:

- **X and Y Position:** Specify screen coordinates for the LOV window in the form coordinate units
- **Width and Height:** Define size of the LOV window in the current form coordinate units; can be resized by the user
- **Column Mapping:** Click the button labeled More to open the LOV Column Mapping window; the slide graphic shows mapping LOV columns to return items on the form.
- **Filter Before Display:** Determines whether the user should be prompted with a dialog box that enables the user to enter a search value before the LOV is invoked; the value will be used as additional restriction on the first column in the query. The graphic in the slide shows the filter applied prior to displaying the LOV.
- **Automatic Display:** Controls whether the LOV should be invoked automatically when form operator enters an item to which the LOV is attached
- **Automatic Select:** Specifies if the LOV should close and return values automatically when reduced to single entry, as illustrated in the slide graphic
- **Automatic Skip:** Determines whether the cursor skips to the next navigable item when the operator selects a value from the LOV to populate the text item, as illustrated in the slide graphic

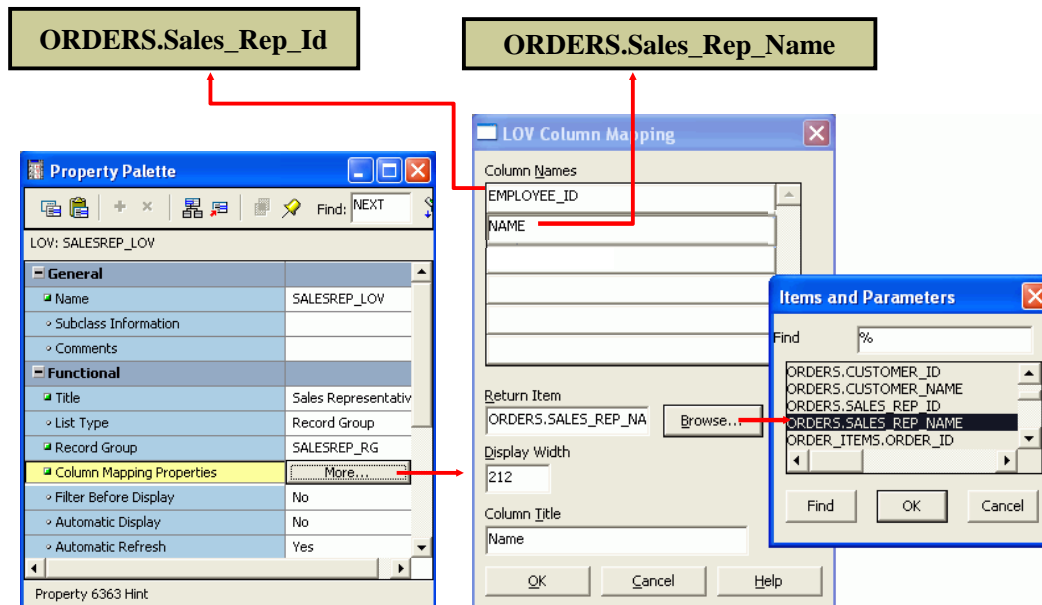
Setting LOV Properties



Setting LOV Properties (continued)

- Automatic Refresh:** If Yes, the record group reexecutes the query every time the LOV is invoked; if No, the query fires only the first time when the LOV is invoked in the session.
Note: You can base more than one LOV on the same record group. When this is the case and you set Automatic Refresh to No, Forms Builder does not reexecute the LOV query after any of the associated LOVs are invoked.
- Automatic Position:** Determines whether Forms automatically positions the LOV near the item from which it was invoked; the slide graphic illustrates this setting by showing the LOV moved close to the text item that invokes it.
- Automatic Column Width:** Determines if Forms automatically sets the width of each column so that the entire title is displayed if the title is longer than the column display width, as illustrated in the graphic

Setting Column Mapping Properties



ORACLE

8 - 17

Copyright © 2009, Oracle. All rights reserved.

Setting Column Mapping Properties

When you click More for Column Mapping Properties, the LOV Column Mapping dialog box opens, with the following elements:

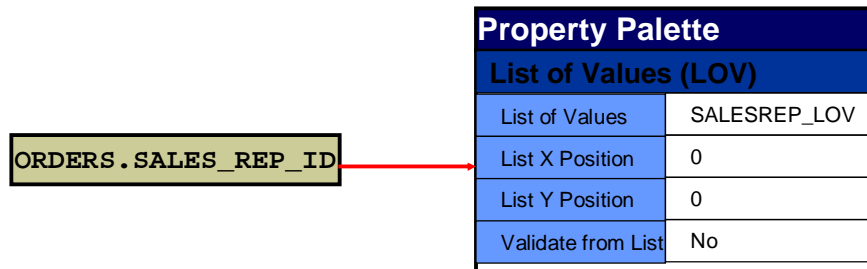
- **Column Names:** Lets you select an LOV column for mapping or defining a column
- **Return Item:** Specifies the name of the form item or variable to which Forms should assign the column value. If null, the column value is not returned from the LOV. If you want to return a value, specify one of the following:
 - `<block_name>.<item_name>`
 - `GLOBAL.<variable_name>`
 - `PARAMETER.<parameter_name>`
- **Display Width:** Width of column display in LOV; value of zero causes the column to be hidden, but value is available for return
- **Column Title:** Heading for column in the LOV window

To set a column mapping, first select the column from the Column Names list, and then set the other mapping values, as required. The slide illustrates mapping the NAME column from the LOV to the ORDERS.Sales_Rep_Name item on the form.

Note: You can modify the record group query from its own properties list. The record group columns and LOV columns must remain compatible.

Associating an LOV with a Text Item

Set List of Values (LOV) properties of the text item.



Associating an LOV with a Text Item

To enable the user to invoke an LOV from a text item, you must specify the LOV name in the Property Palette of the text item.

1. Select the text item in the Object Navigator from which the LOV is to be accessible.
2. In the item Property Palette, set the List of Values (LOV) property to the required LOV.

The slide shows setting the List of Values (LOV) property for the `ORDERS.Sales_Rep_Id` text item to `SALESREP_LOV`. This means that the List of Values lamp is displayed when the user navigates to this text item, indicating that the LOV is available through the List of Values key or menu command.

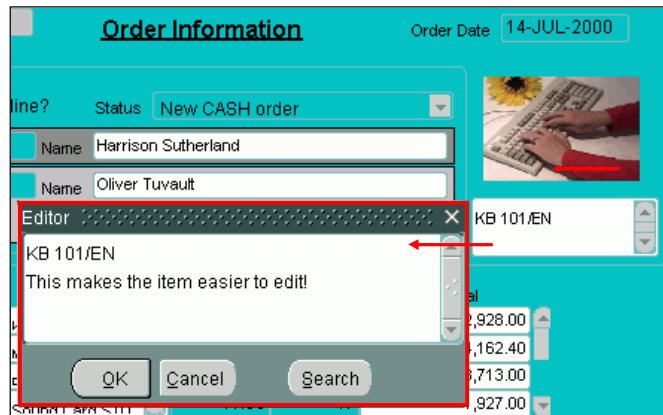
Other List of Values properties for a text item include:

- List X/Y Position: The X and Y coordinates where you want the LOV to display for this particular text item; if set to 0, use the properties that were set for the LOV
- Validate from List: Restrict the valid values for the text item to those that are contained in the LOV

What Are Editors?

Editors:

- Override default editor
- Used for special requirements such as larger editing window, position, color, and title



ORACLE

8 - 19

Copyright © 2009, Oracle. All rights reserved.

What Are Editors?

With a text editor the user can view multiple lines of a text item simultaneously, search and replace text in it, and generally modify the value of an item from this separate window.

Every text item has the default editor available, but you can design your own replacement editor for those items that have special requirements such as larger editing window, position, color, and title.

By overriding the default editor for a text item, you can provide a larger editing window for items with potentially large textual values.

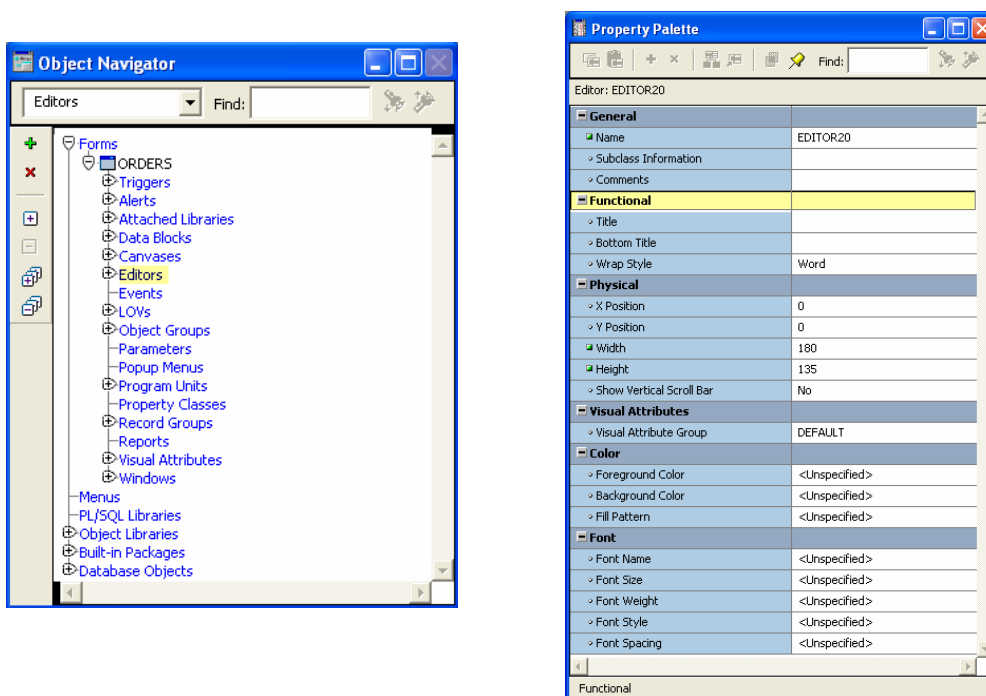
How to Use an Editor at Run Time

With the cursor on the text item to be edited, follow these steps:

1. Click Edit, or select Edit > Edit to invoke the attached editor.
2. Edit the text in the Editor window. Forms Builder editors provide a Search button that invokes an additional search-and-replace dialog box for manipulating text.
3. Click OK to write your changes back to the text item.

The screenshot shows the default editor that is invoked from a multi-line text item. The editor gives the user more space for editing the text item, and also offers the search-and-replace functionality.

Defining an Editor



Defining an Editor

If the user needs to use an editor on text values, the default Forms Builder editor is usually sufficient for most items. However, you can design your own customized editor as an object in a form module, and then attach it to the text items that need it.

How to Create a Customized Editor

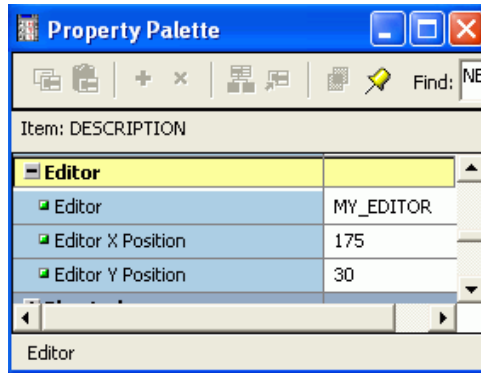
1. Select the Editors node in the Object Navigator, and then click Create, as illustrated in the screenshot at the left of the slide. A new editor object is displayed in the list.
2. Select the new editor in the Object Navigator, and then access its Property Palette, where you can set its name and other properties. The Property Palette for an editor is shown at the right of the slide, with the property categories of General, Functional, Physical, Visual Attributes, Color, and Font.

Setting Editor Properties

The following properties show the individual customization that is possible by creating your own editor:

- **Title/Bottom Title:** Displays at the top or bottom of the editor window
- **Width/Height:** Control size of editor window and, therefore, its editing area
- **X/Y Position:** Screen position for editor; can be overridden by a text item property
- **Wrap Style:** How text wraps in the window: None, Character, or Word
- **Show Vertical Scrollbar:** Specify Yes to add vertical scroll bar to the editor window

Associating an Editor with a Text Item



- Associate one of two types of editors with a text item.
- Set text item's Editor property to one of the following:
 - Null (default Forms Builder editor)
 - Editor name (customized editor)

ORACLE

Associating an Editor with a Text Item

To associate an editor with a text item, you must specify the editor in the Property Palette of the text item.

Select the text item in the Object Navigator from which the editor is to be accessible.

In the item Property Palette, set the Editor property to one of the following settings:

- **Null:** The text item uses the default Forms Builder editor.
- **Editor Name:** The text item uses the named editor that you have created and customized in this module.

The screenshot shows the Property Palette for the Description text item. It shows that the Editor property is set to MY_EDITOR. This specifies that when the user invokes an editor from the Description text item at run time, the MY_EDITOR custom editor is invoked.

Summary

In this lesson, you should have learned that:

- An LOV is a scrollable pop-up window that enables a user to pick the value of an item from a multicolumn dynamic list
- The easiest way to design, create, and associate LOVs with text items is to use the LOV Wizard
- An editor is a separate window that enables the user to view multiple lines of a text item simultaneously, search and replace text in it, and modify the text
- You create editors in the Object Navigator and associate them with text items in the item's Property Palette

ORACLE

8 - 22

Copyright © 2009, Oracle. All rights reserved.

Summary

In this lesson, you learned that lists of values (LOVs) and text editors can be used to support text items. Both LOVs and editors are objects in a form module that open their own window when activated at run time and are used to support text items in any block of the form module.

- LOVs and editors can be shared across text items.
- The steps to implement an LOV are the following:
 1. Create a new LOV (and record group).
 2. Define column mapping for return items.
 3. Attach the LOV to text items, as required.
- The LOV Wizard performs these steps automatically.
- Text items can use the default editor or a user-named editor.

Practice 8: Overview

This practice covers the following topics:

- Creating an LOV and attaching the LOV to a text item
- Creating an editor and attaching it to a text item

ORACLE

8 - 23

Copyright © 2009, Oracle. All rights reserved.

Practice 8: Overview

In this practice session, you will create three LOVs and an editor.

- Using the LOV Wizard, create an LOV in the Orders form to display product numbers and their descriptions. Attach the LOV to the Product_Id item in the ORDER_ITEMS data block.
- Using the LOV Wizard, create an LOV in the Orders form to display sales representatives' IDs and names. Attach the LOV to the Sales_Rep_Id item in the ORDERS data block. Save and run the form.
- Using the LOV Wizard, create an LOV in the Customers form to display sales representatives' numbers and their names. Attach the LOV to the Acct_Mgr_Id item in the CUSTOMERS data block. Save and run the form.
- In the Customers form, create an editor for the Phone_Numbers item.

9

Creating Additional Input Items

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Identify the item types that allow input
- Create a check box
- Create a list item
- Create a radio group

ORACLE

9 - 2

Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

In addition to text items, Oracle Forms Builder provides a variety of other item types. These can be divided into two groups: those that accept input and those that do not. This lesson covers input items and how they are used.

What Are Input Items?

- Item types that accept user input include:
 - Check boxes
 - List items
 - Radio groups
- Input items enable insert, update, delete, and query.

ORACLE

9 - 3

Copyright © 2009, Oracle. All rights reserved.

What Are Input Items?

Input item is a generic term for Forms Builder item types that accept user input.

These item types include the following:

- Check box
- List item
- Radio group

What Can You Do with Input Items?

When you create input items, they already have some initial functionality. Through input items, you can interact with the database in the following ways:

- Insert values.
- Update existing values.
- Delete existing values.
- Query existing values.

Note: You can add functionality to input items with triggers and PL/SQL program units.

What Are Check Boxes?

- Two-state interface object:
 - Checked
 - Unchecked
- Not limited to two values



ORACLE

9 - 4

Copyright © 2009, Oracle. All rights reserved.

What Are Check Boxes?

A *check box* is a two-state interface object that indicates whether a certain value is ON or OFF. The display state of a check box is always either checked (selected) or unchecked (deselected). The screenshots show a check box in both its selected and deselected states.

You can use check boxes to enhance the user interface by converting existing items that have two possible states. Although a check box is limited to two states, it is not limited to just two values. You specify the value to represent checked, the value to represent unchecked, and how other values are processed.

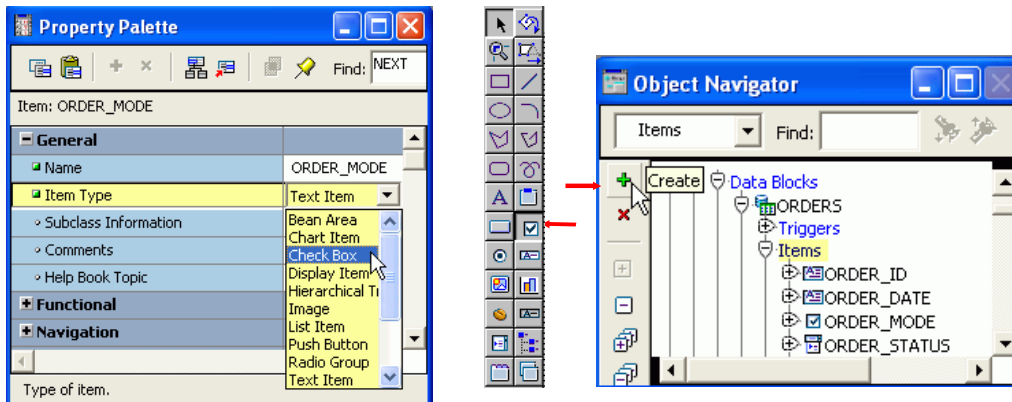
Using a Check Box at Run Time

You can do the following at run time:

- Set check box values either by user input, by means of the Initial Value property, or programmatically.
- In Enter Query mode:
 - Query checked values by clicking one or more times until item is selected
 - Query unchecked values by clicking one or more times until item is deselected
 - Ignore check box values by not selecting or deselecting the initial displayed value

Creating a Check Box

- Convert an existing item.
- Use the Check Box tool in the Layout Editor.
- Use the Create icon in the Object Navigator, then convert the text item to a check box.



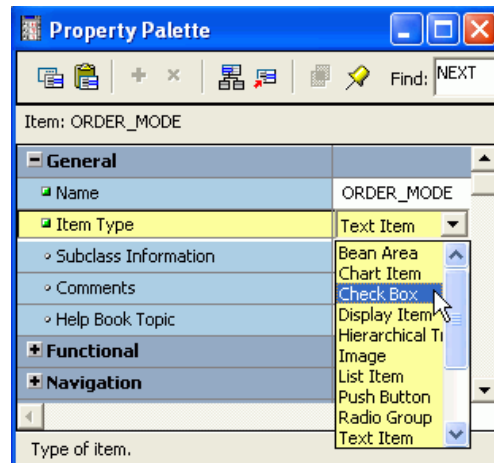
Creating a Check Box

A check box can be created by one of the following methods:

- Converting an existing item—the screenshot at the left shows that the Item Type drop-down list for a text item includes the Check Box item type that you can select.
- Using the Check Box tool in the Layout Editor—the screenshot in the middle of the slide shows the Check Box tool.
- Using the Create icon in the Object Navigator (creates a text item, which you can convert to a check box)—the screenshot at the right of the slide shows creating a text item in the Object Navigator.

Converting an Existing Item to a Check Box

Convert text item
to check box.



ORACLE

9 - 6

Copyright © 2009, Oracle. All rights reserved.

Converting an Existing Item to a Check Box

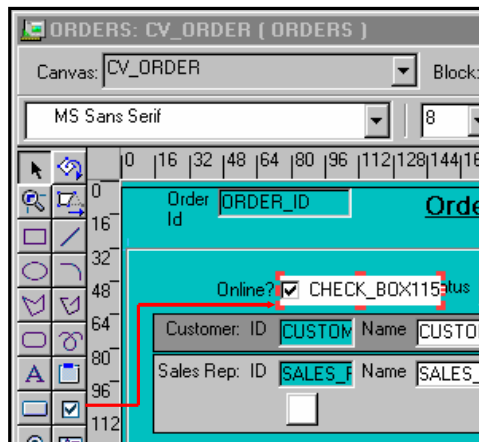
You can convert an existing item to a check box by changing the Item Type property to Check Box in the Property Palette and setting other relevant properties. Perform the following steps:

1. Invoke the Property Palette for the item that you want to convert.
2. Set the Item Type property to Check Box.
3. Specify a check box label.
4. Specify values for the checked and the unchecked states.
5. Set the “Check Box Mapping of Other Values” property.
6. Specify an initial value for the check box item.

Note: The check box label that you specify is displayed to the right of the check box element at run time. If the complete label name is not displayed, adjust it in the Layout Editor. If the item already has a prompt, delete it in the item Property Palette.

Creating a Check Box in the Layout Editor

Use the Check Box tool
in the Layout Editor.



ORACLE

9 - 7

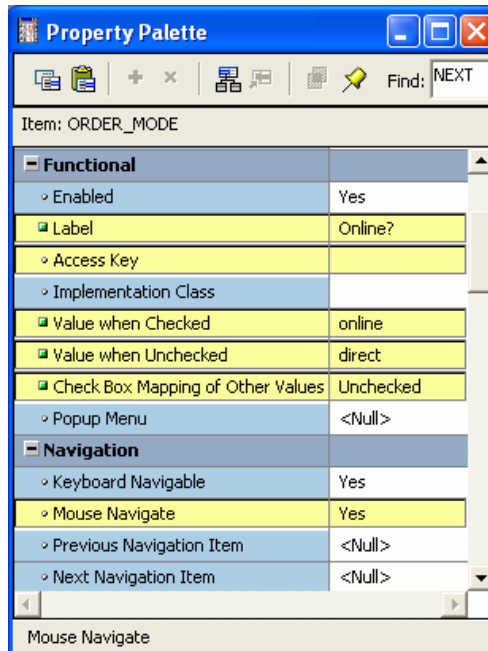
Copyright © 2009, Oracle. All rights reserved.

Creating a Check Box in the Layout Editor

You can also create a check box by using the Check Box tool in the Layout Editor.

1. Invoke the Layout Editor.
2. Set the canvas and block to those on which you want the check box item to be displayed.
3. Click the Check Box tool.
4. Click the canvas in the position where you want the check box to be displayed. The screenshot shows the Layout Editor right after this step has been performed.
5. Double-click the check box to invoke its Property Palette.
6. Set the properties as required.

Setting Check Box Properties



- Data Type
- Label
- Access Key
- “Value when Checked”
- “Value when Unchecked”
- “Check Box Mapping of Other Values”
- Mouse Navigate

ORACLE

9 - 8

Copyright © 2009, Oracle. All rights reserved.

Setting Check Box Properties

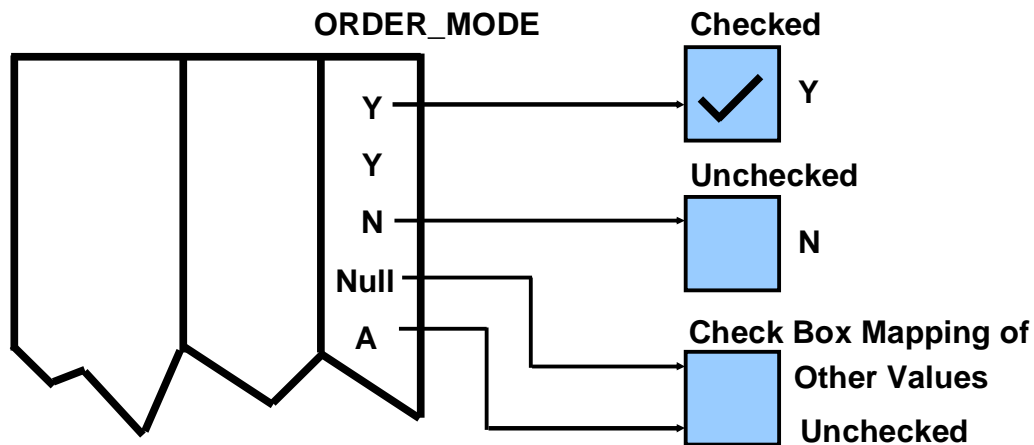
The following properties may be set to affect the appearance and behavior of check boxes:

- **Data Type:** Must be compatible with values specified in the Value properties
- **Label:** Text label displayed next to check box item (independent of the check box value)
- **Access Key:** Which combination of keys may be used to navigate to this item and select or deselect it
- **Initial Value:** Initial value of the item for new record, determining whether check box is initially selected or deselected
- **“Value when Checked”:** Value to represent selected state of the check box
- **“Value when Unchecked”:** Value to represent deselected state of the check box
- **“Check Box Mapping of Other Values”:** How other values are to be processed (Not Allowed, Checked, or Unchecked)
- **Mouse Navigate:** Whether Forms navigates to and moves input focus to the item when the user activates the item with the mouse (default is Yes)

The screenshot shows the Property Palette for a check box for an input item that can have values of either online or direct. The following property values are set:

- **Label:** Online?
- **“Value when Checked”:** online
- **“Value when Unchecked”:** direct
- **“Check Box Mapping of Other Values”:** Unchecked
- **Mouse Navigate:** Yes

Check Box Mapping of Other Values



Check Box Mapping of Other Values

Dealing with Other Values

If your base-table column accepts other values, your check box should account for them. You can assign other values to either the checked or unchecked states by using the Check Box Mapping of Other Values property. Alternatively, you can choose not to accept other values with the Not Allowed setting.

Note: If you choose not to accept other values and they exist in the base-table column, Forms ignores the entire record during query processing.

Dealing with Null Values

If your base-table column accepts null values, you can account for them by one of the following methods:

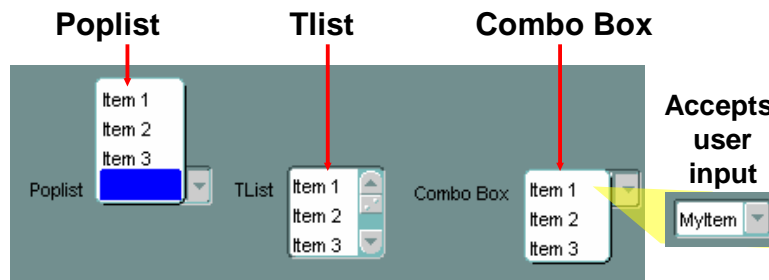
- Set the “Check Box Mapping of Other Values” property.
- Set the checked or unchecked state to represent null (leave the value blank).

The example in the slide shows a database table with a column named **ORDER_MODE** that has values of Y, N, and A, and also some null values. When Checked is set to Y for the check box, Unchecked is set to N, Checkbox Mapping of Other Values is set to Unchecked, all records can be displayed in the form block, but only the Y records show the check box to be selected.

What Are List Items?

List Items:

- Set of mutually exclusive choices, each representing a different value
- Three list styles available:



- Space-saving alternative to a radio group
- Smaller-scale alternative to an LOV

ORACLE

9 - 10

Copyright © 2009, Oracle. All rights reserved.

What Are List Items?

A *list item* is an interface object that displays a predefined set of choices, each corresponding to a specific data value. You use the list item at run time to select a single value. List choices or elements are mutually exclusive; only one can be selected at a time.

List Item Styles

- **Poplist:** It appears as a field with a button attached to the right; click the poplist or its button to display all its list elements.
- **Tlist:** It appears as a rectangular box that displays the list elements; when the display area is not big enough to display all the list elements, a scroll bar is automatically attached to the right to view the remaining list elements, or you can use the up and down arrow keys to navigate the list.
- **Combo Box:** It appears as a field with a drop-down list; click the button to display all the combo box list elements. As illustrated on the slide, the combo box is the only list item type that accepts user input of values not on the list.

What Are List Items? (continued)

Note: The poplist and combo box take up less space, but end users must open them to see the list elements. A tlist remains “open,” and end users can see multiple values at a time. Use the attached scroll bar to see more values if the tlist is not big enough to display all the list elements.

Uses and Benefits of List Items

- Enable display of a defined set of choices.
- Display a set of choices without using a vast area of canvas.
- Provide an alternative to radio groups.
- Provide a Windows-style list of values.

Setting the Value for a List Item

The value for a list item can be set in any of the following ways:

- User selection
- User input (combo box style only)
- A default value
- Programmatic control

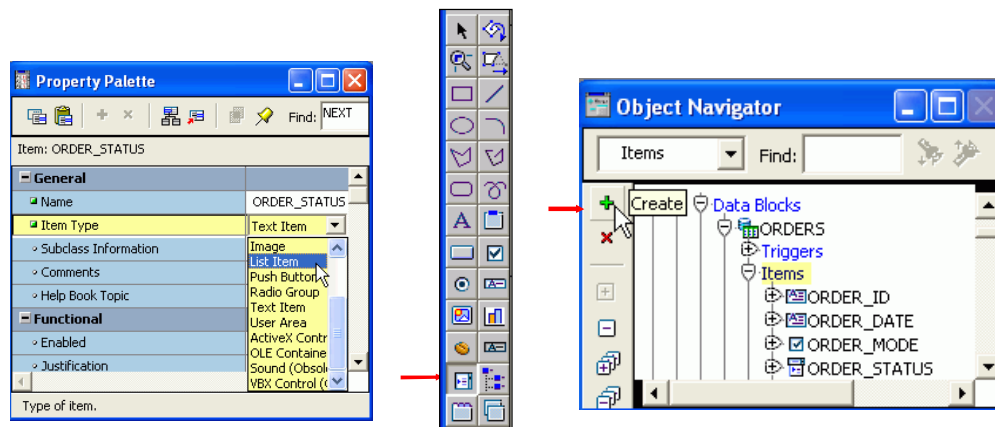
What are some of the differences between a list item and an LOV?

List items:

- Are generally used for a small number of elements
- Do not have a Find button
- Cannot be attached to other items
- Usually have choices that are not based on a `SELECT` statement (although they can be if the list elements are created programmatically)

Creating a List Item

- Convert an existing item.
- Use the List Item tool in the Layout Editor.
- Use the Create icon in the Object Navigator, then convert the text item to a list item.

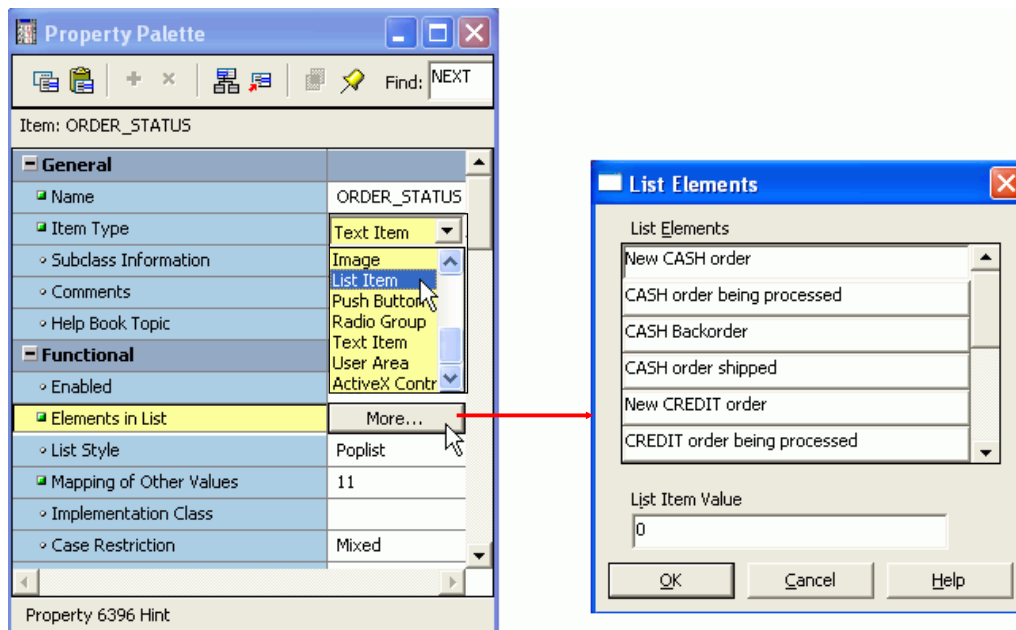


Creating a List Item

A list item can be created by:

- Converting an existing item; the screenshot at the left of the slide shows that the Item Type drop-down list for a text item includes the List Item item type that you can select.
- Using the List Item tool in the Layout Editor; the screenshot at the middle of the slide shows the List Item tool.
- Using the Create icon in the Object Navigator (creates a text item that you can convert to a list item.); the screenshot at the right of the slide shows creating a text item in the Object Navigator.

Converting an Existing Item to a List Item



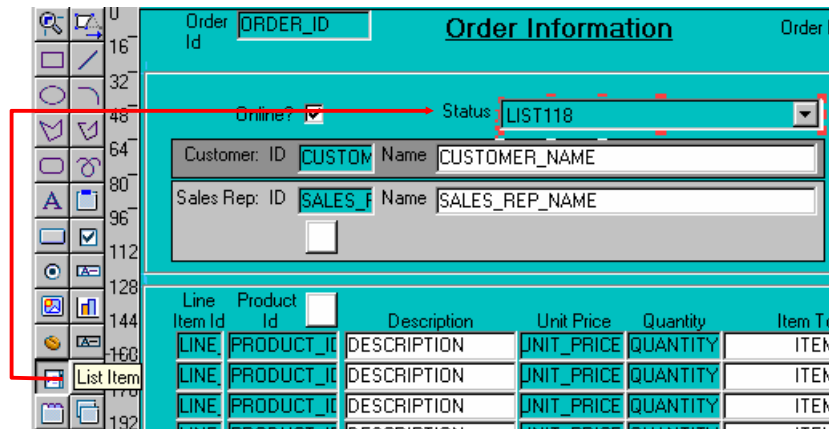
Converting an Existing Item to a List Item

You can convert an existing item to a list item by changing its Item Type property to List Item and setting the relevant properties.

1. Invoke the Property Palette for the item that you want to convert.
2. Set the Item Type property to List Item.
3. Select the “Elements in List” property and click More. The List Elements dialog box appears.
4. Enter the element that you want to display in your list item in the List Elements column; the example shows possible display values for the Order_Status item, such as “New CASH order” and “CASH order being processed”.
5. Enter the value for the currently selected list element in the List Item Value field. The example in the slide shows that if the user selects the displayed value of “New CASH order”, the value 0 is entered in the List Item Value field.
6. Create additional list elements and values by repeating steps 5 and 6.
7. Click OK to accept and close the List Elements dialog box.
8. Set the Other Values property to do one of the following:
 - Reject values other than those predefined as list values.
 - Accept and default all other values to one of the predefined list element values.
9. Enter an initial value for the list item.

Creating a List Item in the Layout Editor

Use the List Item tool
in the Layout Editor.



ORACLE

Creating a List Item in the Layout Editor

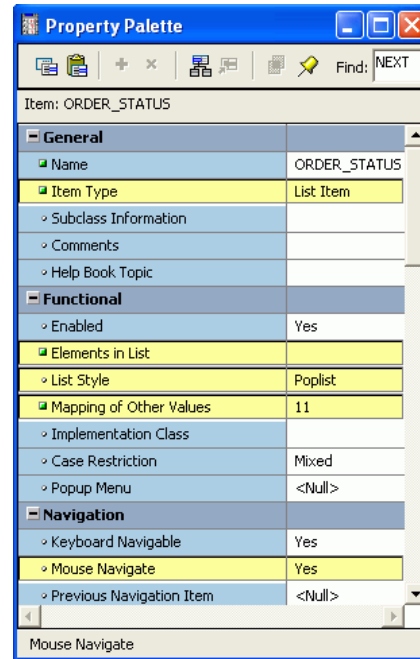
You can also create a list item by using the List Item tool in the Layout Editor.

1. Invoke the Layout Editor.
2. Set the canvas and block to those on which you want the list item to be displayed.
3. Select the List Item tool.
4. Click the canvas in the position where you want the list item to be displayed. The screenshot shows the Layout Editor just after this step has been performed.
5. Double-click the list item to invoke its Property Palette.
6. Set the properties as required.

Note: To obtain a list of available functions when defining list elements, press Ctrl + K while the input focus is in the List Elements window. The Keys window may appear behind the List Elements window; if so, just move the List Elements window so that you can see the Keys window.

Setting List Item Properties

- “Elements in List”:
 - List elements
 - List item value
- List Style
- “Mapping of Other Values”
- Mouse Navigate

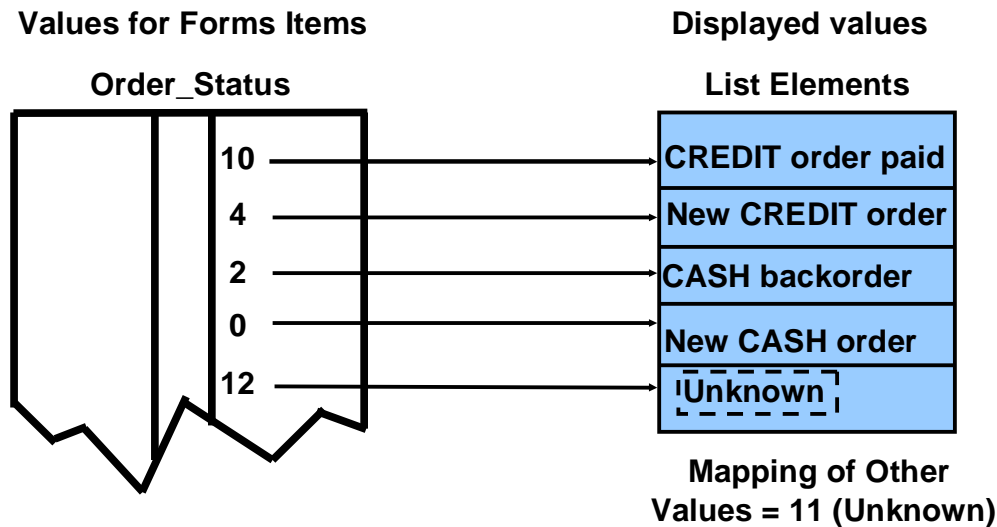


Setting List Item Properties

- **“Elements in List”**: Clicking More opens the List Elements dialog box, where you specify:
 - List Elements: List elements that display at run time
 - List Item Value: Actual value that corresponds to the list element
- **List Style**: Display style of list item (Poplist, Tlist, or Combo Box)
- **“Mapping of Other Values”**: How other values are processed
- **Mouse Navigate**: Whether Forms navigates to the item and moves input focus to it when the user clicks the item

The screenshot shows the Property Palette for the Order_Status list item with List Style set to Poplist, “Mapping of Other Values” set to 11 (which in this application corresponds to a status of “Unknown”), and Mouse Navigate set to Yes.

List Item Mapping of Other Values



ORACLE

9 - 16

Copyright © 2009, Oracle. All rights reserved.

List Item Mapping of Other Values

NULL Values in a List Item

If the base-table column for a list item accepts NULL values, Forms Builder creates a pseudochoice in the list to represent the null.

All three list styles display a blank field if a query returns a NULL value. If the Data Required property is set to No:

- Poplist displays a blank element for a NULL value
- The user can omit a selection for TList or can click Clear Field to deselect all list elements. This sets the list item to NULL.
- Combo Box does not display a blank element. The end user must delete the default value if the default value is not NULL.

Handling Other Values in a List Item

If the base-table column for a list item accepts values other than those associated with your list elements, you must specify how you want to handle the values. Do this in one of the following ways:

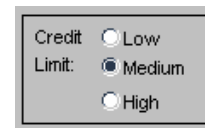
- Ignore other values by leaving the "Mapping of Other Values" property blank.
- Associate the other values with one of the existing list elements (by naming either the list element or its associated value) in the "Mapping of Other Values" property.

List Item Mapping of Other Values (continued)

In the example in the slide, order status codes were defined to include 0 through 11. The table, however, contains a record with the status code of 12. For the list item in the form, the “Mapping of Other Values” property is set to 11 (meaning status is unknown.) The form then displays the record with the status code of 12, but displays its status as unknown.

What Are Radio Groups?

- Set of mutually exclusive radio buttons, each representing a value
- Use:
 - To display two or more static choices
 - As an alternative to a list item
 - As an alternative to a check box



What Are Radio Groups?

A *radio group* is an item where a set of radio buttons represents the possible values for the item. These values and hence their corresponding radio buttons are mutually exclusive.

Uses and Benefits of Radio Groups

Radio groups provide:

- A choice between two or more static values; the screenshot shows a radio group for Credit Limit with choices of Low, Medium, or High.
- An alternative to list items that have only a few choices
- A choice between two alternatives, where choice is not On/Off or Yes/No (for example, Landscape or Portrait print format)

Note: Consider list items instead of radio groups if there are more than four or five choices.

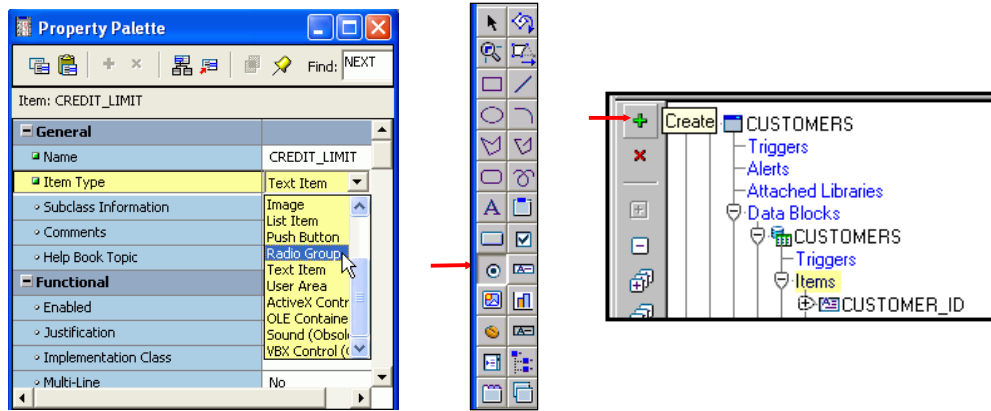
Using a Radio Group at Run Time

You can do the following at run time:

- Set radio group values:
 - By user input
 - By means of the Initial Value property
 - Programmatically
- Query individual radio button values.

Creating a Radio Group

- Convert an existing item.
- Create a new radio button in the Layout Editor, which implicitly creates a radio group also.
- Use the Create icon in the Object Navigator, then convert the text item to a radio group.

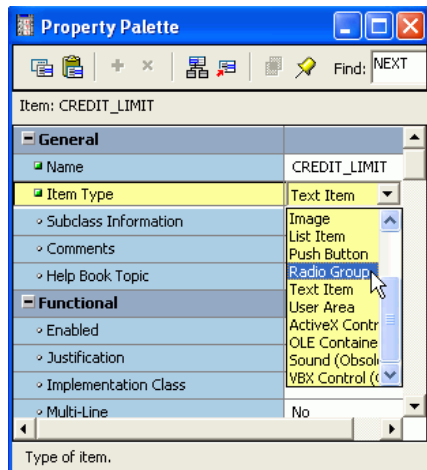


Creating a Radio Group

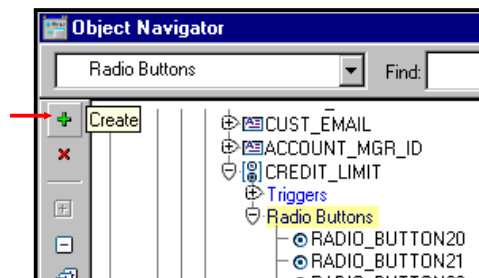
A radio group can be created by:

- Converting an existing item to a radio group; the screenshot at the left of the slide shows that the Item Type pop-up list for a text item includes the Radio Group item type that you can select.
- Creating a new radio button in the Layout Editor (automatically creates a radio group if none exists); the screenshot in the middle of the slide shows the Radio Button tool.
- Using the Create icon in the Object Navigator (this creates a text item that you can convert to a radio group); the screenshot at the right shows creating a text item in the Object Navigator.

Converting an Existing Item to a Radio Group



Change Item Type and set other properties.



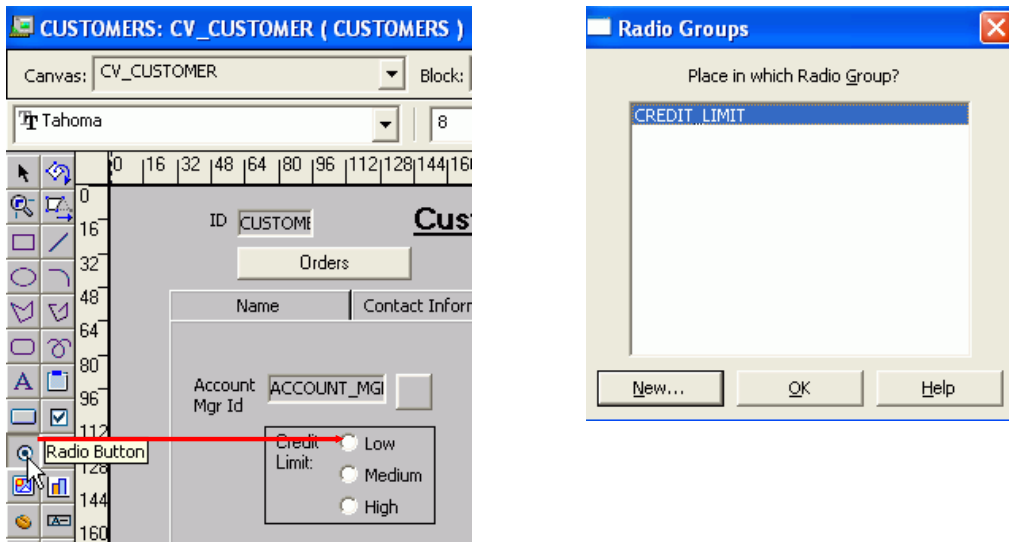
Create radio buttons for the radio group.

Converting an Existing Item to a Radio Group

You can convert an existing item to a radio group by changing the item type and setting the properties for a radio group.

1. Invoke the Property Palette for the item that you want to convert.
2. Set the Item Type property to Radio Group.
3. Set the Canvas property to the Canvas on which you want the radio buttons to appear.
4. Set the “Mapping of Other Values” property to specify how the Radio Group should handle any other values.
5. Set the Initial Value property, as required. This should be the name of a radio button.
6. Expand the item node in the Object Navigator.
The Radio Buttons node appears.
7. Select the Radio Buttons node and click the Create icon. A radio button displays in the Object Navigator and the Property Palette takes on its context.
8. Enter a name, value, and a label for the radio button.
9. Specify the display properties of the radio button.
10. Create additional radio buttons by repeating steps 6 through 9.

Creating a Radio Group in the Layout Editor



Creating a Radio Group in the Layout Editor

You can also create a radio group by using the Radio Button tool in the Layout Editor.

1. Invoke the Layout Editor.
2. Set the canvas and block to those on which you want the radio group to be displayed.
3. Click the Radio Button tool.
4. Click the canvas at the desired location for the radio group, as shown in the screenshot at the left of the slide.

If you already have a radio group in the current block, the Radio Groups dialog box appears and you must decide whether the new radio button should appear in the existing group or a new one. If you select New, the new radio group is created implicitly. The screenshot at the right of the slide shows the new radio button being created in the existing Credit_Limit radio group.

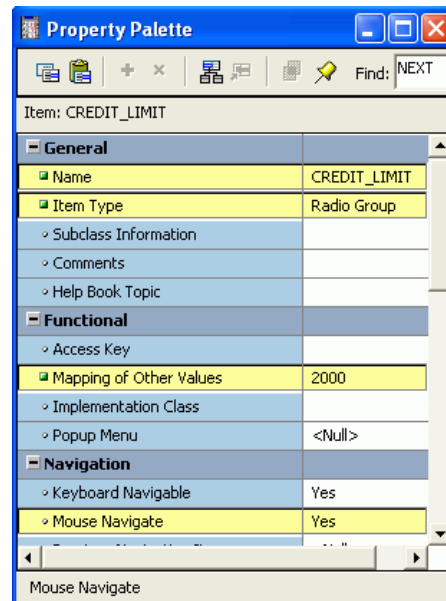
5. Double-click the radio button to invoke its Property Palette.
6. Set the radio button properties as required.

Note: The canvas property for a radio group is set in the Property Palette of the radio group. The individual radio buttons do not have a canvas property.

Setting Radio Group Properties

Radio group properties:

- “Mapping of Other Values”
- Data Type
- Initial Value
- Mouse Navigate



Setting Radio Group Properties

You should set the following properties for radio groups:

- **Data Type:** Must be compatible with “Mapping of Other Values” for the Radio Group, and Radio Button Value for each Radio Button in the group
- **“Mapping of Other Values”:** How values other than those specified are processed
- **Mouse Navigate:** Whether Forms navigates to the item when the operator clicks the item

You also must specify a valid Initial Value, except under either of the following conditions:

- The radio group accepts other values.
- The value associated with one of the radio buttons is NULL.

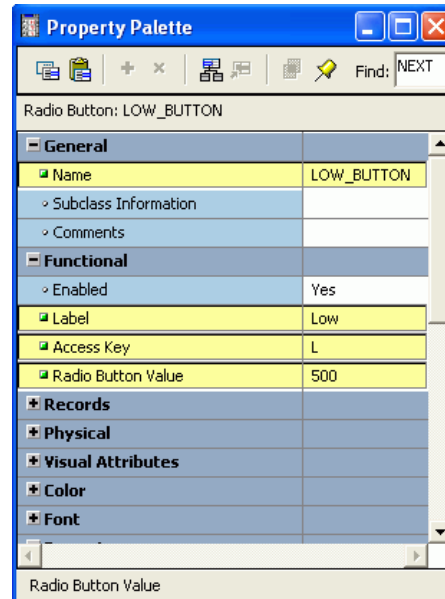
The screenshot shows setting the following property values for the Credit_Limit radio group:

- **“Mapping of Other Values”:** 2000
- **Mouse Navigate:** Yes

Setting Radio Button Properties

Radio button properties:

- Label
- Access Key
- Radio Button Value



Setting Radio Button Properties

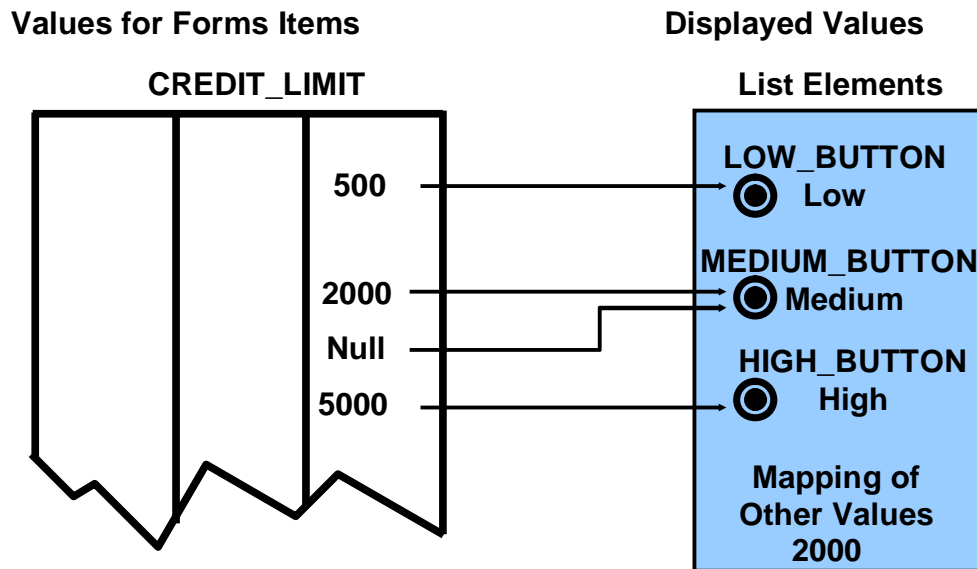
You should set the following properties for radio buttons:

- **Label:** Text that appears adjacent to the radio button (independent of the button value)
- **Access Key:** Which combination of keys can be used to navigate to and manipulate this radio button
- **Radio Button Value:** The value for this item if this radio button is selected

The screenshot shows setting the following property values for the Low_Button radio button:

- **Label:** Low
- **Access Key:** L
- **Radio Button Value:** 500

Radio Group Mapping of Other Values



ORACLE

9 - 24

Copyright © 2009, Oracle. All rights reserved.

Radio Group Mapping of Other Values

Handling Other Values in a Radio Group

If the base-table column for a radio group accepts values other than those associated with your radio buttons, you must use one of the following methods to specify how you want to handle the values:

- Ignore other values by leaving the radio group's "Mapping of Other Values" property blank.
- Associate the other values with one of the existing radio buttons by naming the associated value of the button in the "Mapping of Other Values" property.

Note: Ignoring other values results in the entire row being ignored during query processing.

NULL Values in a Radio Group

A radio group can treat NULL as a valid value. You should account for the NULL case, if your base-table column allows null values. You can do this in one of the following ways:

- Use the "Mapping of Other Values" property to implicitly force NULL to a radio button.
- Assign the NULL to its own radio button by leaving the Radio Button Value property blank.

The example in the slide shows that when "Mapping of Other Values" is set to 2000, null values appear with the radio button labeled Medium selected.

Summary

In this lesson, you should have learned that:

- Check boxes, list items, and radio groups are the item types that allow input
- You create these items by:
 - Changing the item type of an existing item
 - Using the appropriate tool in the Layout Editor
- You can use a check box for items that have only two possible states
- You can use a list item to enable users to pick from a list of mutually exclusive choices
- You can use a radio group for two or three mutually exclusive alternatives

ORACLE

9 - 25

Copyright © 2009, Oracle. All rights reserved.

Summary

In this lesson, you learned how to create items that accept direct user input. Use these items to enhance the user interface:

- **Check boxes:** To convert items that have two possible states
- **List items (Poplists, Tlists, and Combo Boxes):** To convert items that have mutually exclusive choices
- **Radio groups:** To convert items with two or three alternatives that are mutually exclusive

Practice 9: Overview

This practice covers the following topics:

- Converting a text item into a list item
- Converting a text item into a check box item
- Converting a text item into a radio group
- Adding radio buttons to the radio group

ORACLE

9 - 26

Copyright © 2009, Oracle. All rights reserved.

Practice 9: Overview

In this practice, you convert existing text items into other input item types. You create a list item, a check box, and a radio group.

- In the Orders form, convert the Order_Status item into a list item. Save and run the form.
- In the Orders form, convert the Order_Mode item into a check box item.
- In the Customers form, convert the Credit_Limit item into a radio group. Add three radio buttons in the radio group. Save and run the form.

10

Creating Noninput Items

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Identify item types that do not allow input
- Create a display item
- Create an image item
- Create a button
- Create a calculated item
- Create a hierarchical tree item
- Create a bean area item

ORACLE

10 - 2

Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

Some Oracle Forms item types do not accept user input (noninput items), but they do provide an effective means of accessing data and initiating actions. This lesson describes how to create and use noninput items.

Noninput Items: Overview

Item types that do not accept direct user input include:

- Display items
- Image items
- Push buttons
- Calculated items
- Hierarchical tree items
- Bean area items

ORACLE

10 - 3

Copyright © 2009, Oracle. All rights reserved.

Noninput Items: Overview

“Noninput items” is a generic term for item types that do not accept direct user input. However, you can set the value of some noninput items by programmatic control. Noninput items can be divided into two groups—those that can display data and those that cannot.

- Noninput items that can display data:
 - Display items
 - Image items
 - Calculated items
 - Hierarchical tree items
- Noninput items that cannot display data:
 - Push buttons

Bean area items can fall into either of these categories, depending on the functionality of the implemented JavaBean.

Use noninput items to enhance your application by displaying additional data, often from a nonbase table.

What Are Display Items?

Display items:

- Are similar to text items
- Cannot:
 - Be edited
 - Be queried
 - Be navigated to
 - Accept user input
- Can display:
 - Non-base-table information
 - Derived values



ORACLE

10 - 4

Copyright © 2009, Oracle. All rights reserved.

What Are Display Items?

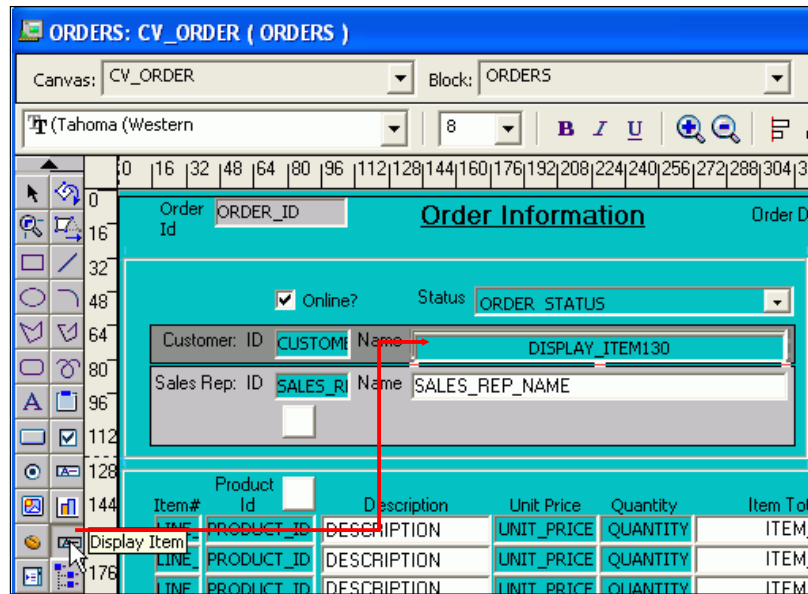
A display item is similar to a text item, except that it cannot be edited or navigated to at run time. A display item is a read-only text box whose value must be fetched or assigned programmatically.

You can use display items to display:

- Additional, non-base-table information
- Derived data values

The screenshot shows two display items, the customer name and the sales representative name, that were created by using the Display Item tool in the Layout Editor.

Creating a Display Item



ORACLE

10 - 5

Copyright © 2009, Oracle. All rights reserved.

Creating a Display Item

A display item can be created by using:

- The Layout Editor
- The Create icon in the Object Navigator (creates a text item that you can convert to a display item)
- The Item Type property to convert an existing item into a display item

To create a display item in the Layout Editor, perform the following steps:

1. Invoke the Layout Editor.
2. Display the desired canvas and select the correct data block.
3. Click the Display Item tool.
4. Click the canvas at the position where you want to place the display item. The screenshot shows the Layout Editor after this step is performed.
5. Double-click the new display item to display its Property Palette.
6. Change the Name from `DISPLAY_ITEMXX` to the desired name.
7. Specify the other properties as needed. You can set the required item properties in the Property Palette:
 - Set Database Item to No for a nonbase table display item.
 - You can assign a format mask for a single-line display item by setting its Format Mask property.

Image Items

- Use image items to display images.
- Images imported from a file system are stored in any supported file type.
- Images imported from a database are stored in a LONG RAW column or a BLOB column.



Line Item Id	Product Id	Description	Unit Price	Quantity	Item Total
1	2289	KB 101/ES	46	200	9,200.00

ORACLE

10 - 6

Copyright © 2009, Oracle. All rights reserved.

Image Items

Using Images

You can use images as graphic objects within a form module. A graphic image is displayed automatically and cannot be manipulated at run time. It can be imported from the database or the file system.

Alternatively, you can display images within image items.

What Is an Image Item?

An image item is a special interface control that can store and display vector or scanned images. Just as text items store and display VARCHAR2, number, or date values, image items store and display images.

Like other items that store values, image items can be either data block items or control items.

The screenshot shows an image item that displays a product image for a selected product in an order. The Layout Editor tool for creating an image item is also shown.

Image Items (continued)

Displaying Image Items

You can populate an image item by using one of the following methods:

- Fetching from a LONG RAW or BLOB database column
- Using a trigger and a built-in to populate the image item programmatically; you learn about triggers in later lessons.

Storing Images

You can store images in either the database or the file system.

When you insert images into the database by means of a save (commit), they are automatically compressed using Oracle image compression.

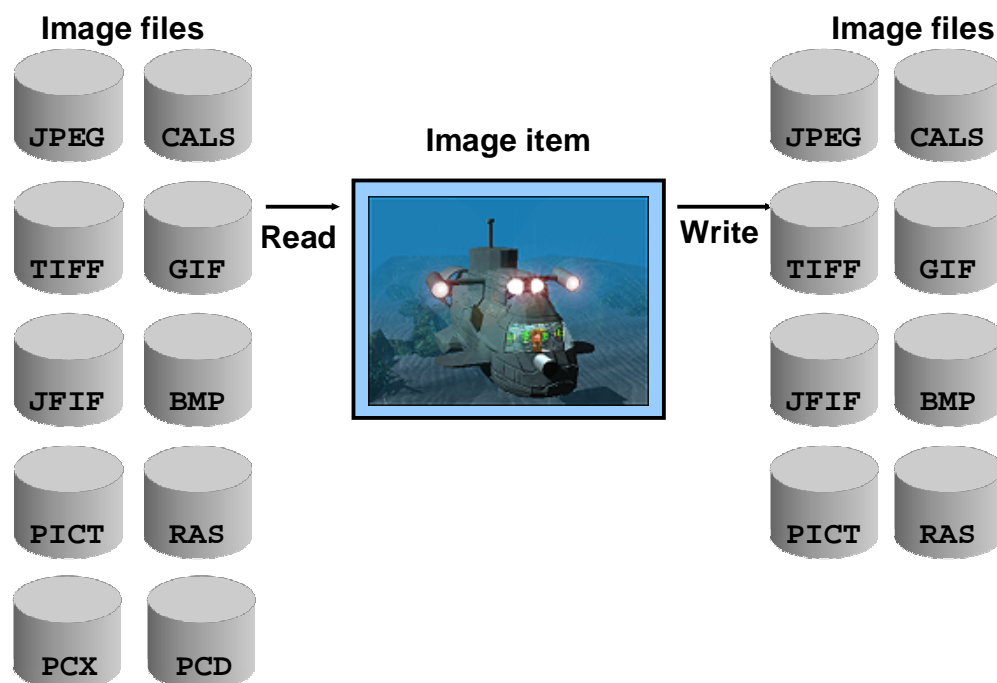
Where Image Is Stored	Description
Database	LONG RAW column compressed image that can be up to 2 gigabytes or BLOB column that can be up to four gigabytes
File	Any supported file format

Note: To conserve application server memory when displaying large image items, reduce the number of records that are buffered by manipulating the “Number of Records Buffered” data block property.

Technical Note

You can also populate an image item with a BFILE, but you need to use DBMS_LOB to do so.

Image File Formats



ORACLE

10 - 8

Copyright © 2009, Oracle. All rights reserved.

Image File Formats

Forms Builder supports the following image formats:

File Suffix	Description	Image Items
BMP	MS Windows and OS/2 BitMap Picture	Read/Write
CALS	CALS type raster	Read/Write
GIF	Graphics interchange format	Read/Write
JFIF	JPEG file interchange format	Read/Write
TIFF (single page)	Tag image file format	Read/Write
JPEG	Joint Photographic Experts Group	Read/Write
PICT	Macintosh Quickdraw Picture	Read/Write
RAS	Sun Raster	Read/Write
PCX	PC Paintbrush Bitmap Graphic	Read only
PCD	Photo-CD Image	Read only

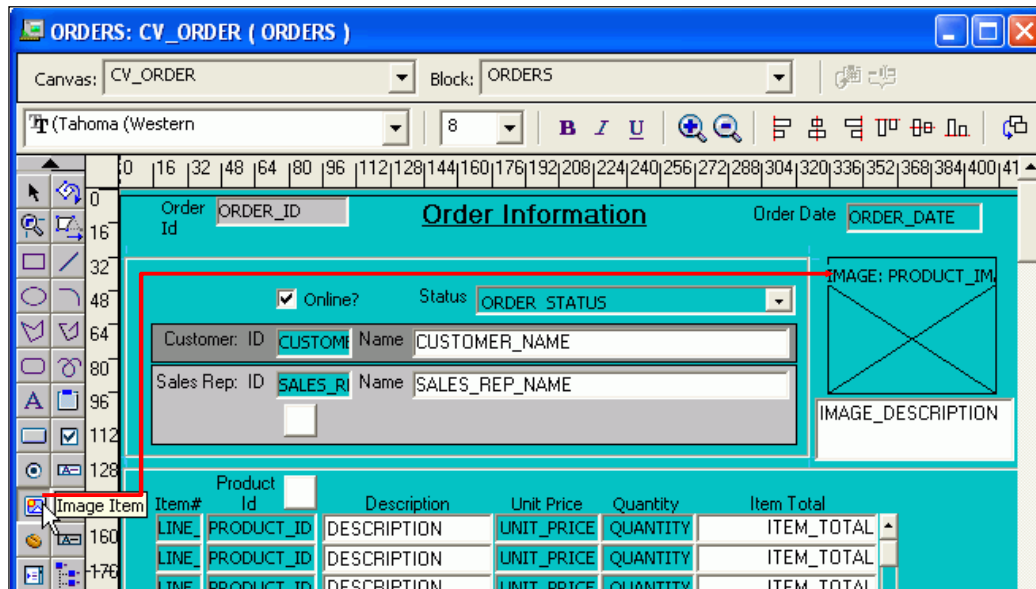
To reduce network traffic, limit the number of image items and background images that must be downloaded from the application server. You can deploy them as Java Archive (JAR) files to reduce network traffic, or you can download them in an alternative manner.

Image File Formats (continued)

For example, to display a company logo in your application, you can include the image in the HTML page that downloads at application startup rather than retrieving the image from the database or the file system. The HTML page can be cached.

The graphics in the slide convey the fact that most image types can be both read into an image item from the database and written from an image item to the database; only `.pcd` and `.pcx` image types are read-only.

Creating an Image Item



ORACLE

10 - 10

Copyright © 2009, Oracle. All rights reserved.

Creating an Image Item

There are three ways to create an image item:

- By using the Image Item tool in the Layout Editor (as described in the next section)
- By using the Create icon in the Object Navigator (creates a text item that you can convert to an image item)
- By converting an existing item into an image item

Steps to Create an Image Item from the Layout Editor

1. Invoke the Layout Editor.
2. Set the canvas and block to those where you want the item to display.
3. Click the Image Item tool.
4. Click the canvas at the position where you want the image item to display. The screenshot in the slide shows the Layout Editor after this step has been performed.
5. Double-click the image item. The Property Palette appears.
6. Change the name from IMAGEXX to the desired name.
7. Specify the other properties as needed.

Note: Remember to set the Database Item property to No for an image item whose value is not stored in the base table.

Setting Image-Specific Item Properties

- Image Format
- Image Depth
- Display Quality
- Sizing Style
- Show Horizontal Scroll Bar
- Show Vertical Scroll Bar

Functional	
◦ Enabled	Yes
◦ Image Format	TIFF
◦ Image Depth	Original
◦ Display Quality	High
◦ Sizing Style	Crop
◦ Popup Menu	<Null>
Navigation	
Data	
Records	
Database	
Physical	
◦ Visible	Yes
◦ Canvas	CANVAS2
◦ Tab Page	<Null>
◦ X Position	68
◦ Y Position	68
◦ Width	72
◦ Height	72
◦ Bevel	Lowered
◦ Show Horizontal Scroll Bar	No
◦ Show Vertical Scroll Bar	No

ORACLE

10 - 11

Copyright © 2009, Oracle. All rights reserved.

Setting Image-Specific Item Properties

The screenshot shows the Property Palette of an image item with the default property values.

Set the following properties to affect the appearance and behavior of the image item:

- **Image Format:** Format in which image is stored in the database (default is TIFF)
- **Image Depth:** Depth setting for image being read from or written to a file on the file system (Original, Monochrome, Gray, LUT , or RGB) (default is Original)
- **Display Quality:** Resolution used to display the image item; controls trade off between quality and performance (High, Medium, or Low) (default is High)
- **Sizing Style:** Determines how much of the image is displayed when the sized image does not match the size of the item (Crop cuts off edges of the image so that it fits in item display area; Adjust scales the image so that it fits within the item display area.) (default is Crop)
- **Show Horizontal/Vertical Scroll Bar:** Displays scroll bars to enable scrolling of the image that does not fit into the item display area (default is No)

Note: Image items do not have a Data Type property. If you set an image item Database Item property to Yes, Forms Builder understands that the data type is LONG RAW.

What Are Push Buttons?

Push buttons:

- Cannot display or represent data
- Are used to initiate an action
- Display as:

- Text button



- Icon



ORACLE

10 - 12

Copyright © 2009, Oracle. All rights reserved.

What Are Push Buttons?

A *push button* is an interface object that you click to initiate an action. A push button is usually displayed as a rectangle with a descriptive label inside. Push buttons cannot store or display values.

You can enhance your form module by adding push buttons to provide quick and direct access to the most needed operations.

Push Button Styles

Forms Builder supports two push button styles:

- **Text button:** Displayed with a text label on the push button (a button with the Exit label in the screenshot)
- **Icon:** Displayed with a bitmapped graphic on the push button and often used in toolbars (a button with an icon of an exit door in the screenshot in the slide)

Push Button Actions

Use buttons to:

- Move input focus
- Display a list of values (LOV)
- Invoke an editor
- Invoke another window
- Commit data
- Issue a query
- Perform calculations

ORACLE

10 - 13

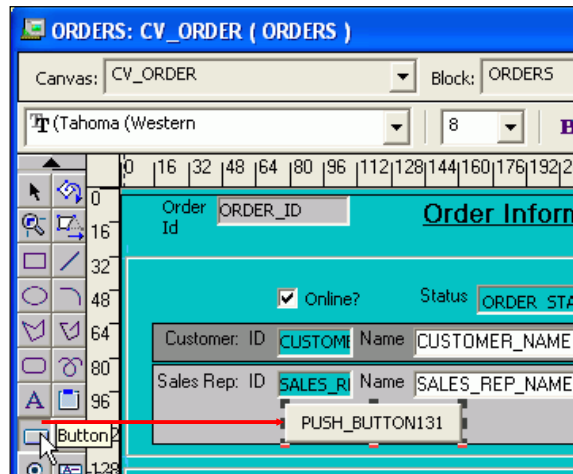
Copyright © 2009, Oracle. All rights reserved.

Push Button Actions

You use push buttons to enable users to perform some type of action. Clicking the button invokes code that is contained in a When-Button-Pressed trigger, as described in the lesson titled “Producing Triggers.” There is no default action performed by a button; you must code the action that you want a button to perform.

Note: Push buttons do not accept input focus on some window managers. On these platforms, the Keyboard Navigable property has no effect, and users can interact with the items only by using a mouse. Clicking a push button does not move the input focus on these platforms. The input focus remains in the item that was active before the user clicked the push button.

Creating a Push Button



ORACLE

10 - 14

Copyright © 2009, Oracle. All rights reserved.

Creating a Push Button

A push button can be created by using:

- The Push Button tool in the Layout Editor
- The Create icon in the Object Navigator (creates a text item that you can convert to a push button)

How to Create a Push Button from the Layout Editor

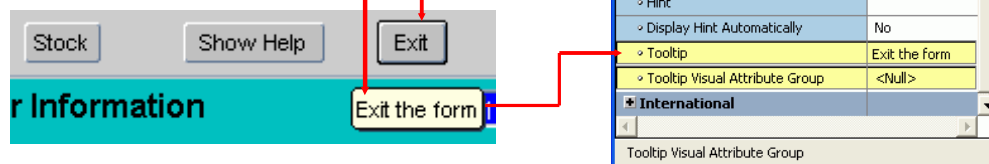
To create a push button from the Layout Editor, perform the following steps:

1. Invoke the Layout Editor.
2. Set the canvas and block to those where you want the push button to display.
3. Click the Push Button tool.
4. Click the canvas at the position where you want the push button to display. The screenshot shows the Layout Editor immediately after this step has been performed.
5. Double-click the push button. The Property Palette appears.
6. Change the name from PUSH_BUTTONXX to the required name.
7. Specify the other properties as required.

Note: Icon image files for buttons must be in GIF or JPEG format for run time, but can be .ico files to display at design time.

Setting Push Button Properties

- Label
- Iconic
- Icon Filename
- Default Button
- Mouse Navigate
- Tooltip
- Tooltip Visual Attribute Group



10 - 15

Copyright © 2009, Oracle. All rights reserved.

ORACLE

Setting Push Button Properties

The slide shows the Property Palette of a button and how it appears at run time. Some of the properties to set for a button include:

- **Label:** Text label that appears on the button at run time; the example shows an Exit label
- **Iconic:** Indicates whether the button displays an icon instead of a label; example is no icon
- **Icon Filename:** Name of the file that contains the icon resource (file name only without the extension, such as list, not list.gif)
- **Default Button:** Indicates whether this is the default push button for the block, which can be selected implicitly by pressing [Enter] without the need to navigate or use the mouse (The default push button may be bordered or highlighted in a unique fashion to distinguish it from other push buttons. The screenshot of the run-time form, at the bottom left of the slide, shows the Exit button with a distinct border, indicating that it is the default button.)
- **Mouse Navigate:** Indicates whether Forms navigates to the item when you click it
- **Tooltip:** Help text that should appear in a tool tip beneath the button when the cursor moves over it; screenshots in the slide show a tool tip of “Exit the form”
- **Tooltip Visual Attribute Group:** Named visual attribute to apply to the tool tip at run time; the example in the slide shows no visual attribute applied

What Are Calculated Items?

Calculated items:

- Accept item values that are based on calculations
- Are read-only
- Can be expressed as:
 - Formula:
 - A calculated item value is the result of a horizontal calculation.
 - It involves bind variables.
 - Summary:
 - A calculated item value is a vertical calculation.
 - A summary is performed on values of a single item over all rows in a block.

ORACLE

10 - 16

Copyright © 2009, Oracle. All rights reserved.

What Are Calculated Items?

With a calculated item, you can declaratively base item values on calculations involving one or more variable values. For example, you can use a calculated item to display a running total of employees' total compensation.

Any item that can store a value can be used as a calculated item by setting certain property values. However, calculated items are read-only, so you should generally use display items as calculated items.

Calculation Modes

Calculations can be expressed as a formula or as a summary of all items in a block. Forms Builder supports the following calculation modes:

- **Formula:** The calculated item value is the result of a horizontal calculation involving one or more bind variables, such as form items, global variables, and parameters. For example, the screenshot on the next slide shows the settings for Item_Total.
- **Summary:** The calculated item value is a vertical calculation involving the values of a single item over all rows within a single block. For example, the screenshot on the next slide also shows the settings for Order_Total.

Setting Item Properties for the Calculated Item

- Formula:

- Calculation Mode
- Formula

Order Total	
Calculation Mode	Formula
Formula	:order_items.quantity * :order_items.unit_price

- Summary:

- Calculation Mode
- Summary Function
- Summarized Block
- Summarized Item

Item Total	
Calculation Mode	Summary
Formula	
Summary Function	Sum
Summarized Block	<Null>
Summarized Item	ITEM_TOTAL

Line	Product	Description	Unit Price	Quantity	Item Total
Item Id	Id				
1	3106	KB 101/EN	48	6	2,928.00
2	3114	MB - S900/650+	96.8	43	4,162.40
3	3123	PS 220V JD	79	47	3,713.00
4	3129	Sound Card STD	41	47	1,927.00
Order Total					46,257.00

ORACLE

10 - 17

Copyright © 2009, Oracle. All rights reserved.

Setting Item Properties for the Calculated Item

Unlike the other items covered so far in this lesson, there is no Item Type property value called Calculated Item. The calculation-specific properties of an item make it a calculated item. Text items and display items both support calculated items.

A calculated item can be created by:

- Setting the calculation-specific properties of any existing item that can store a value
- Creating a new item in the Layout Editor and setting its calculation-specific properties
- Using the Create icon to create a new item and setting calculation-specific properties

The following properties are specific to calculated items:

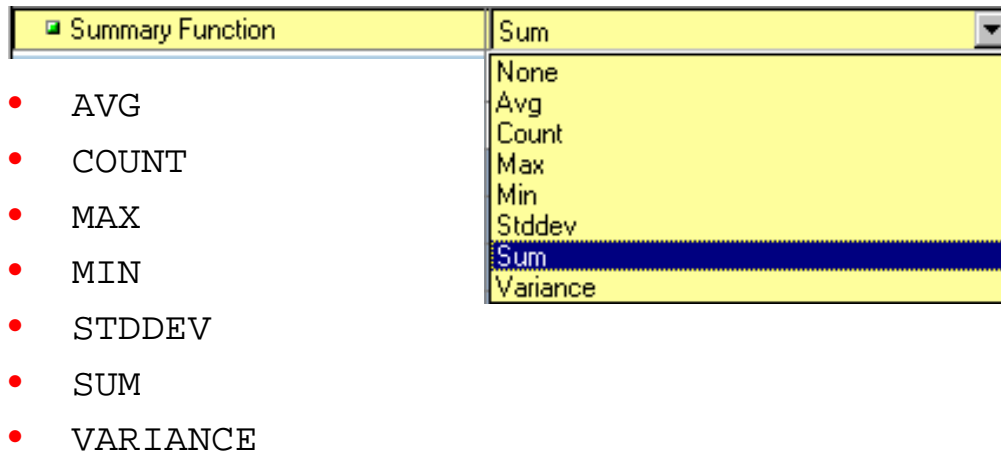
- **Calculation Mode:** The method of computing the calculated item values (None, Formula, or Summary)
- **Formula:** A single PL/SQL expression that determines the calculated value of a formula calculated item; can compute a value or call a subprogram
- **Summary Function:** The type of summary function (discussed on the following page) to be performed on the summary calculated item
- **Summarized Block:** The block over which all rows are summarized in order to assign a value to the summary calculated item; if not specified, current block is assumed
- **Summarized Item:** The item whose value is summarized in order to assign a value to the calculated item; required if the Calculation Mode is Summary

Setting Item Properties for the Calculated Item (continued)

The screenshots in the slide show the calculation properties in Property Palettes of two calculated items, Item_Total (a formula) and Order Total (a summary):

Property	Setting for Item Total	Setting for Order Total
Calculation Mode	Formula	
Formula	:order_items.quantity * :order_items.unit_price	
Summary Function		Sum
Summarized Block		<Null>
Summarized Item		ITEM_TOTAL

Using Summary Functions



ORACLE

10 - 19

Copyright © 2009, Oracle. All rights reserved.

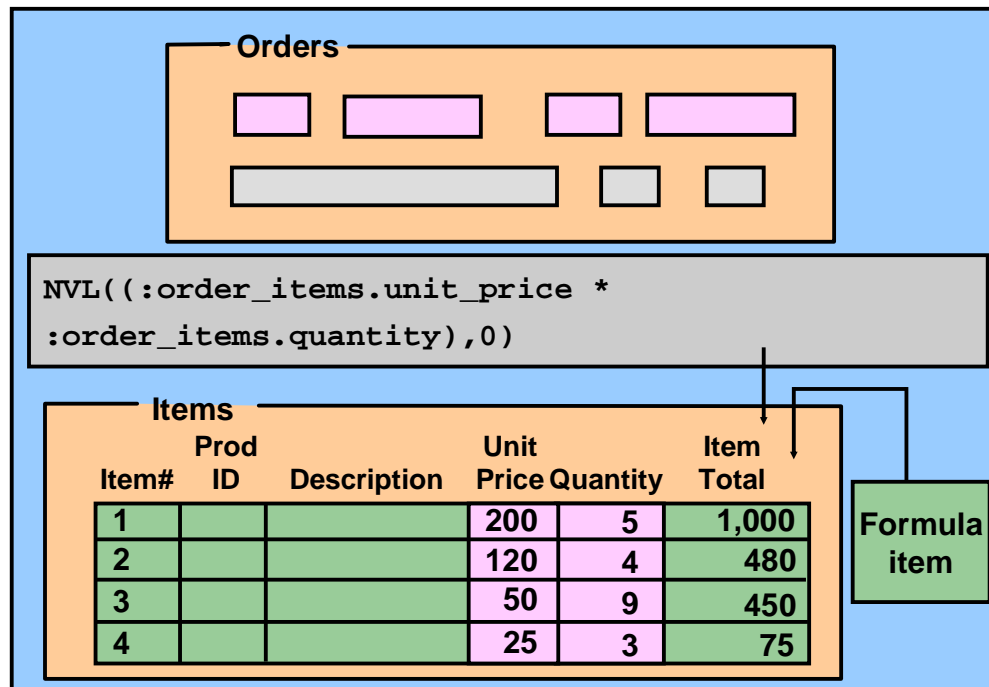
Using Summary Functions

You can use the following standard SQL aggregate functions for summary items:

- **AVG:** The average value (arithmetic mean) of the summarized item over all records in the block
- **COUNT:** Count of all non-NULL instances of the summarized item over all records in the block
- **MAX/MIN:** Maximum/minimum value of the summarized item over all records in the block
- **STDDEV:** The standard deviation of the summarized item's values over all records in the block
- **SUM:** Sum of all values of the summarized item over all records in the block
- **VARIANCE:** The variance (square of the standard deviation) of the summarized item's values over all records in the block

The screenshot shows that the Property Palette displays a drop-down list for the Summary Function property from which you can select one of these functions.

Creating a Calculated Item Based on a Formula



ORACLE

10 - 20

Copyright © 2009, Oracle. All rights reserved.

Creating a Calculated Item Based on a Formula

To create a calculated item based on a formula, perform the following steps:

1. Create a new item in the Object Navigator or the Layout Editor.
2. Open the Property Palette of the item.
3. Set the Calculation Mode property to Formula.
4. Click More for the Formula property and enter the PL/SQL expression to define the formula.

The slide graphic depicts setting a formula for the Item Total. Notice that Forms item names in the formula are preceded with a colon.

Note: A formula item cannot be a database item because its value is computed by Forms, not queried from a database column.

Rules for Calculated Item Formulas

Create calculated item formulas according to the following rules:

- A formula item must not invoke restricted built-ins.
- A formula item cannot execute any data manipulation language (DML) statements.
- Do not terminate a PL/SQL expression with a semicolon.
- Do not enter a complete PL/SQL assignment expression.

ORACLE

10 - 21

Copyright © 2009, Oracle. All rights reserved.

Rules for Calculated Item Formulas

When writing formulas for calculated items, observe the following rules:

- The formula (and any user-written subprogram that calls it) must not invoke any restricted built-ins.
- The formula (and any user-written subprogram that calls it) cannot execute any DML statements.
- Do not terminate the PL/SQL expression with a semicolon.
- If the PL/SQL expression involves an assignment, do not enter the complete PL/SQL statement. Forms Builder assigns the actual assignment code internally.

Example

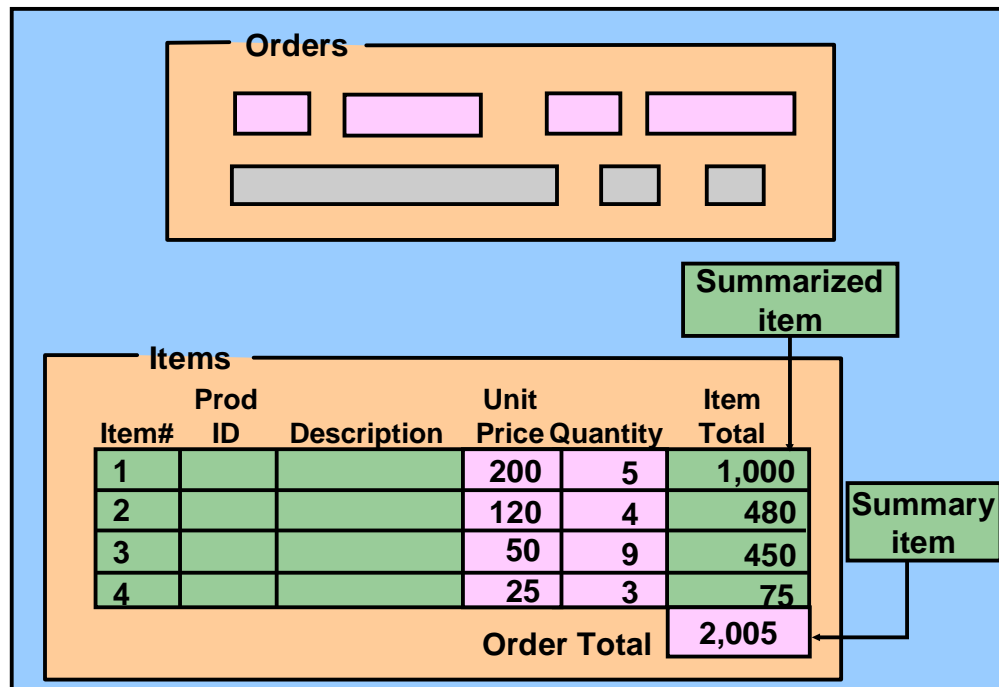
Suppose you have a formula item called `Gross_Comp` in the `EMPLOYEES` block of a form. If you set the Formula property to:

```
NVL(:employees.salary,0) * NVL(:employees.commission_pct,0)
```

Forms Builder internally converts this expression to a complete statement:

```
:employees.gross_comp := (NVL(:employees.salary,0) *  
    NVL(:employees.commission_pct,0));
```

Creating a Calculated Item Based on a Summary



10 - 22

Copyright © 2009, Oracle. All rights reserved.

ORACLE

Creating a Calculated Item Based on a Summary

To create a calculated item based on a summary, perform the following steps:

1. Create a new item in the Object Navigator or the Layout Editor.
2. Open the Property Palette of the item.
3. Set the Calculation Mode property to Summary.
4. Select a function from the Summary Function pop-up list.
5. From the Summarized Block pop-up list, select a block over which all rows will be summarized.
6. From the Summarized Item pop-up list, select an item to be summarized.

The slide graphic depicts setting a summary for the Order_Total. The summarized item is Item_Total. In other words, the order total is the sum of all of the item totals for this order.

Note: A *summary item* is the calculated item to which you assign a value.

A *summarized item* is the item whose values are summarized and then assigned to the summary item.

Rules for Summary Items

- Summary items must reside in either:
 - The same block as the summarized item
 - Or
 - A control block with the Single Record property set to Yes
- Summarized item must reside in either:
 - A control block
 - Or
 - A data block with either the Query All Records property or the Precompute Summaries property set to Yes
- Data Type of summary item must be Number, unless using MAX or MIN.

ORACLE

10 - 23

Copyright © 2009, Oracle. All rights reserved.

Rules for Summary Items

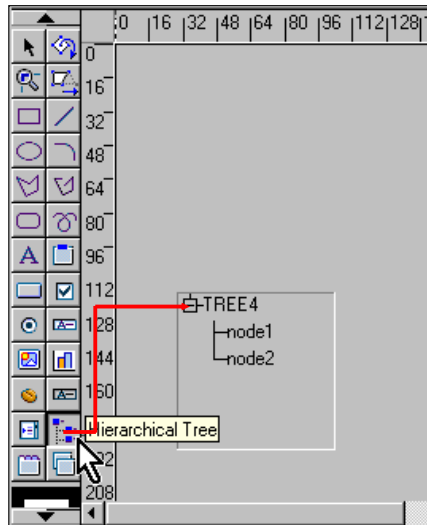
When creating calculated items based on a summary, observe the following rules:

- The summary item must reside in the same block as the summarized item, or in a control block whose Single Record property is set to Yes.
- The summarized item must reside in a control block, or in a data block whose Query All Records property or the Precompute Summaries property is set to Yes.

Note: This ensures that records fetched in the block and the summarized value are consistent. Otherwise, another user may possibly update a record that has not been fetched yet.

- Set the Data Type property for a summary item to Number, unless the summary function is Max or Min, in which case the data type must mirror its associated summarized item. For example, a calculated item that displays the most recent (maximum) date in the HIRE_DATE column must have a data type of Date.
- If the summarized item values are based on a formula, the summarized item must reside in a block whose Query All Records property is set to Yes.

Creating a Hierarchical Tree Item



ORACLE

10 - 24

Copyright © 2009, Oracle. All rights reserved.

Creating a Hierarchical Tree Item

A hierarchical tree is an item that displays data in the form of a standard navigator.

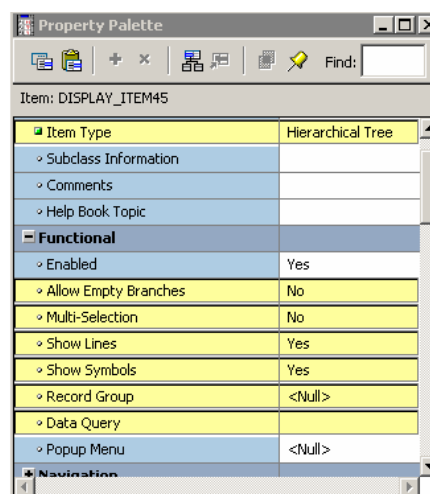
How to Create a Hierarchical Tree Item

To create a hierarchical tree item, do one of the following:

- **In the Layout Editor:**
 - Click the Hierarchical Tree icon.
 - Click and drag on the canvas to create the hierarchical tree object. The screenshot shows the Layout Editor immediately after this step has been performed.
 - Set other hierarchical tree–related properties as required.
- **In the Object Navigator:**
 - Create a new item by using the Create icon.
 - Open the item’s Property Palette and set the Item Type property to Hierarchical Tree.
 - Set other hierarchical tree–related properties as required.

Setting Hierarchical Tree Item Properties

- Allow Empty Branches
- Multi-Selection
- Show Lines
- Show Symbols
- Record Group
- Data Query



ORACLE

Setting Hierarchical Tree Item Properties

Hierarchical Tree properties include the following (as you see in the screenshot of the Property Palette showing the default property values for a hierarchical tree):

- **Item Type:** Must be set to Hierarchical Tree
- **Allow Empty Branches:** Indicates whether branch nodes may exist with no children (if set to the default value of No, branch nodes with no children will be converted to leaf nodes; if set to Yes, an empty branch will be displayed as a collapsed node)
- **Multi Selection:** Indicates whether multiple nodes may be selected at one time (default No)
- **Show Lines:** Indicates whether a hierarchical tree displays lines leading up to each node (default Yes)
- **Show Symbols:** Indicates whether a hierarchical tree should display the + or – symbol in front of each branch node (default Yes)
- **Record Group:** Name of the record group from which the hierarchical tree derives its values (default <Null>)
- **Data Query:** Specifies the query-based data source (default blank)

Several built-ins are available to manipulate hierarchical trees. For more information, see the lesson titled “Adding Functionality to Items,” which discusses how to implement a hierarchical tree in a form.

Note: A hierarchical tree must be the only item in the data block.

What Are Bean Area Items?

The bean area item enables you to:

- Add a JavaBean to a form
- Extend Forms functionality
- Interact with the client machine
- Reduce network traffic



ORACLE

10 - 26

Copyright © 2009, Oracle. All rights reserved.

What Are Bean Area Items?

A JavaBean is a component written in Java that can plug in to any applet or Java application. The bean area item enables you to extend Forms functionality by adding a JavaBean to your form. The slide graphic, a cup of coffee surrounded by beans, symbolizes JavaBeans.

With JavaBeans, you can interact with the client machine, performing such functions as:

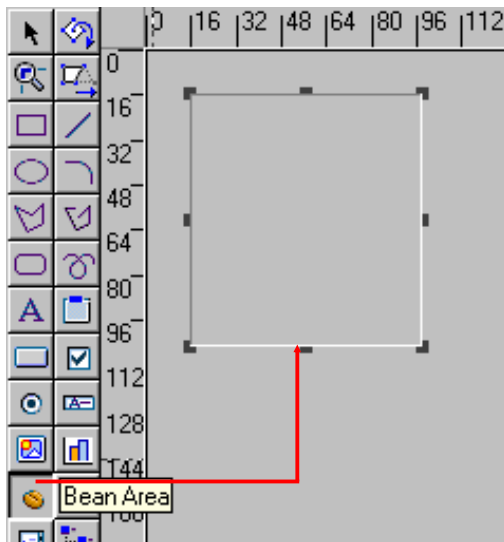
- Obtaining information about the client machine
- Uploading client files to the server machine
- Modifying the user interface on the client
- Checking the spelling of a text item
- Displaying a progress bar, clock, calendar, or color picker with which the operator may be able to interact and select values

Some of this functionality, such as the calendar, is possible using Forms native functionality. However, using a JavaBean enables client interaction without generating network traffic.

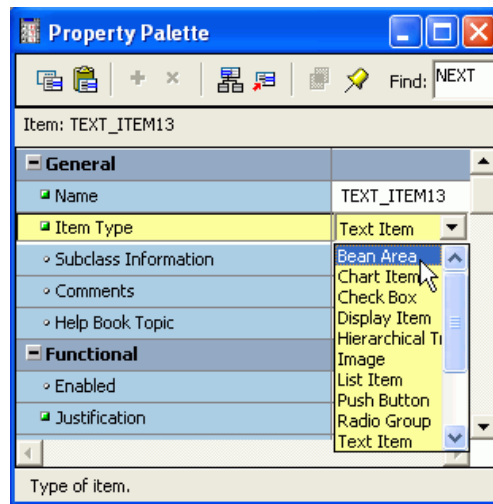
Although JavaBeans can be used to input data, as in the case of the Calendar JavaBean, the bean area item itself does not accept user input.

Creating a Bean Area Item

Create a bean area
in the Layout Editor.



Convert an existing item
to a bean area.



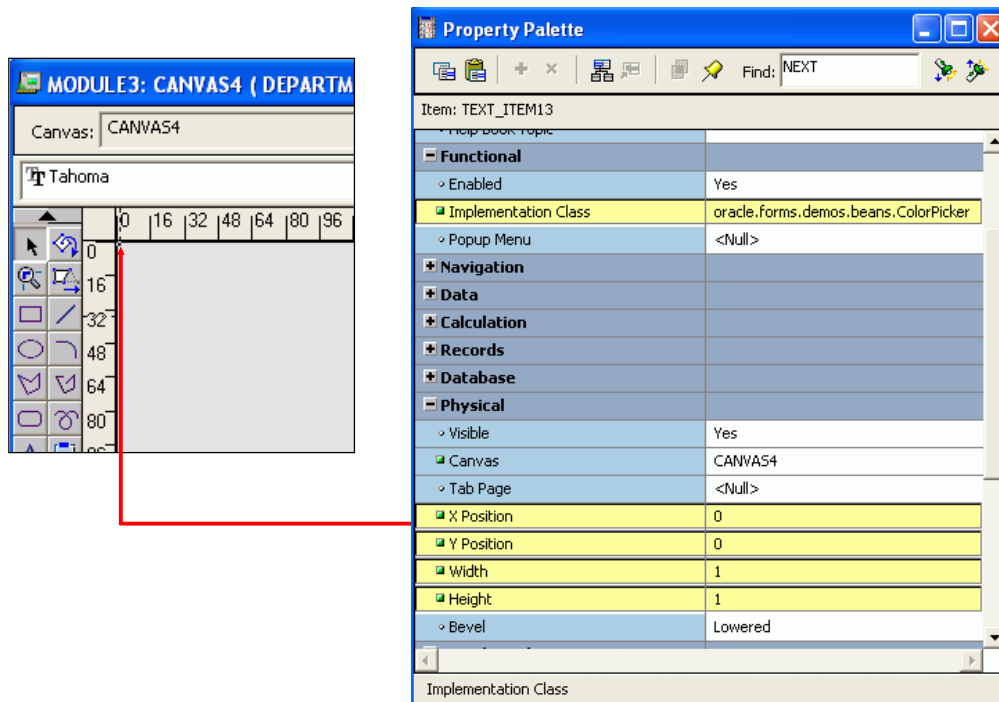
ORACLE

Creating a Bean Area

You create a bean area using the Bean Area tool in the Layout Editor. Click the tool and drag out an area on the canvas. At first the area will look like an empty rectangle, as you see in the screenshot at the left of the slide.

You can also create a bean area from any existing item by changing its item type to Bean Area in the Property Palette. The screenshot at the right of the slide shows that Bean Area is one of the selections from the Item Type drop-down list in the Property Palette for an item.

Setting Bean Area Item Properties



Setting Bean Area Item Properties

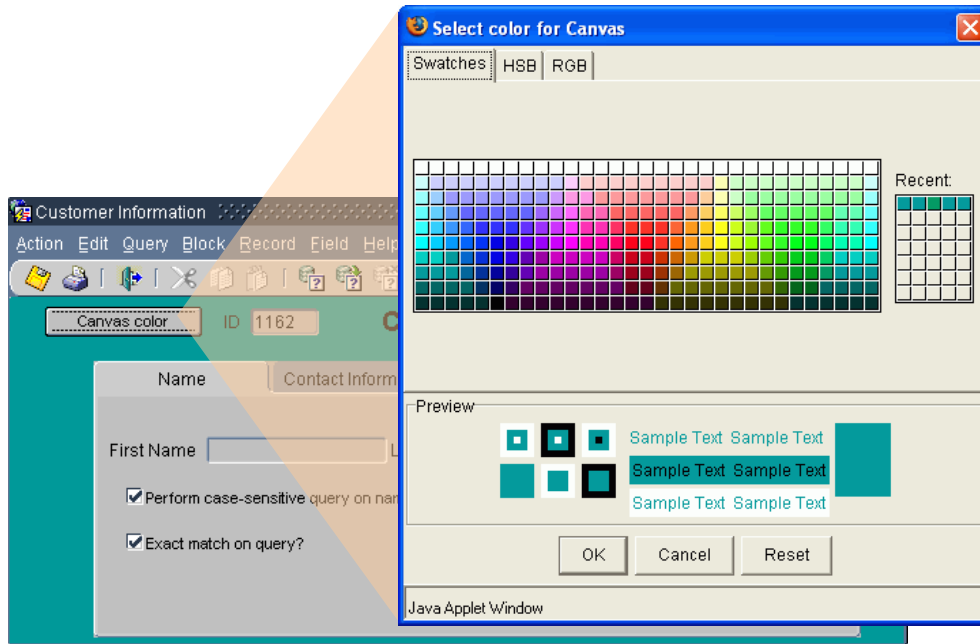
The most important bean area property is Implementation Class. This property is used to specify the fully qualified name of the Java class that the bean area item should instantiate.

It is not necessary to set the Implementation Class at design time. Instead, you can programmatically register the bean at run time. This is discussed in the lesson titled “Adding Functionality to Items.”

If the JavaBean has a visible component, Forms Builder displays the JavaBean within the bean area item in the Layout Editor after the Implementation Class is set. You can set the size and position of the bean area visually in the Layout Editor, or by changing the X/Y Position, Width, and Height properties.

Some JavaBeans have no visible component. To avoid the distraction of a large empty rectangle, you can set the properties so that the bean area is displayed as a tiny dot in the corner of the canvas. This is depicted by the screenshots in the slide, where Width and Height are set to 1 and the bean area shows up in the Layout Editor as a dot. Alternatively, you can set its Visible property to No, which ensures that the bean area does not display at run time, although you still see it in the Layout Editor.

Interacting with the JavaBean at Run Time



Interacting with the JavaBean at Run Time

You may programmatically define functionality for the JavaBean to perform. At run time the user may interact with the JavaBean to initiate this functionality.

In the example in the slide, there is a button that invokes the ColorPicker JavaBean, which displays to the user a color picker window. When the operator selects a color from the color picker, it is used to programmatically set the background color of the canvas.

Programming responses to user interaction is discussed in the lesson titled “Adding Functionality to Items.”

Summary

In this lesson, you should have learned that:

- The following item types do not allow input:
 - Display items (to show non-base-table information)
 - Image items (to display an image)
 - Push buttons (to initiate action)
 - Calculated items (to display the results of a calculation)
 - Hierarchical tree items (to display related data in a hierarchical fashion)
 - Bean area items (to execute client-side Java code)
- You create noninput items by:
 - Changing the type of an existing item and setting certain properties
 - Using the appropriate tool in the Layout Editor

ORACLE

10 - 30

Copyright © 2009, Oracle. All rights reserved.

Summary

In this lesson, you should have learned that:

- Display items display non-base-table information or derived values
- Image items store and display vector or scanned bitmapped images
- Push Buttons initiate an action
- Calculated items base item values on calculations. Calculations can be expressed in one of the following modes:
 - Formula
 - Summary
- Hierarchical trees display information in an Object Navigator style display
- Bean Area items enable you to integrate Java components into your application
- You create the above item types by:
 - Using the appropriate tool in the Layout Editor
 - Clicking Create in the Object Navigator (this creates a text item that you can convert to a different item type)
 - Changing the item type and/or setting appropriate properties on existing items

Practice 10: Overview

This practice covers the following topics:

- Creating display items
- Creating an image item
- Creating buttons with icons
- Creating calculated items:
 - Formula
 - Summary
- Creating a bean area item

ORACLE

10 - 31

Copyright © 2009, Oracle. All rights reserved.

Practice 10: Overview

In this practice, you add several items in the Customers and Orders forms: display items, image item, push buttons, calculated items, and a bean area.

- In the Orders form:
 - Create two display items in the ORDER_ITEMS block.
 - Create an image item in the CONTROL block.
 - Create a button that displays an icon in the CONTROL block.
 - Base the Item_Total item in the ORDER_ITEMS block on a formula.
 - Create a control item in the CONTROL block. Base this item value on a summary that displays the total value of an order.
- In the Customers form:
 - Create an iconic button in the CONTROL block.
 - Create a bean area.
- Save and run the Orders and Customers forms.

The calculated items function immediately because their functionality is defined declaratively. However, the other item types that you create in this practice do not function until programmatic logic is added in later lessons.

11

Creating Windows and Content Canvases

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe the relationship between windows and content canvases
- Create windows and content canvases
- Display a form module in multiple windows
- Display a form module on multiple layouts

ORACLE

11 - 2

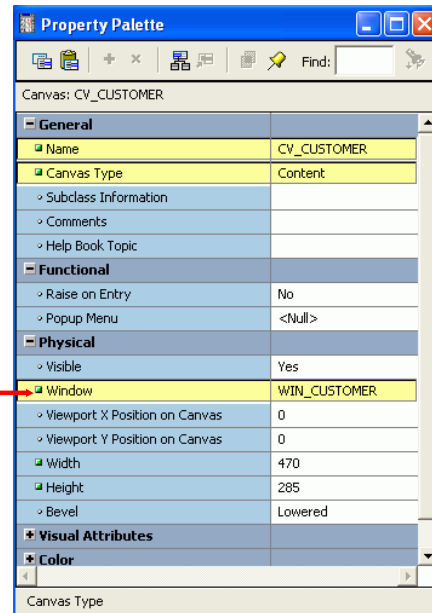
Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

With Oracle Forms Builder, you can take advantage of the graphical user interface (GUI) environment by displaying a form module across several canvases and in multiple windows. This lesson familiarizes you with the window object and the default canvas type, the content canvas.

What Are Windows, Canvases, and Viewports?

- Window: Container for Forms Builder visual objects
- Canvas: Surface on which you “paint” visual objects; to see a canvas and its objects, display the canvas in a window.
- Viewport: Part of the canvas visible in the window



What Are Windows, Canvases, and Viewports?

With Forms Builder, you can display an application in multiple windows by using its display objects—windows and canvases.

Window

A window is a container for all visual objects that make up a Forms application. It is similar to an empty picture frame. The window manager provides the controls for the window that enable such functionality as scrolling, moving, and resizing. You can minimize a window. A single form may include several windows.

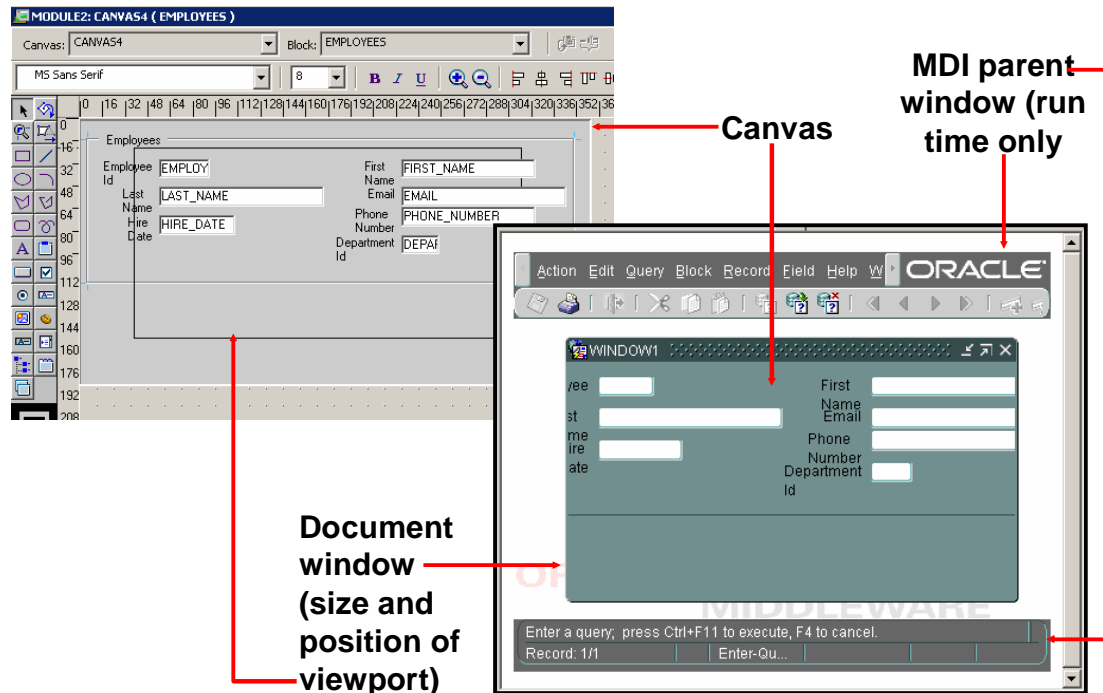
Canvas

A canvas is a surface inside a window container on which you place visual objects, such as interface items and graphics. It is similar to the canvas on which a picture is painted. To see a canvas and its contents at run time, you must display it in a window. A canvas always displays in the window to which it is assigned.

Viewport

A viewport is an attribute of a canvas. It is effectively the visible portion of, or view onto, the canvas. When you resize a viewport, the window in which the canvas is displayed is resized.

Comparing Design Time with Run Time



Comparing Design Time with Run Time

The slide shows windows and canvases at design time and at run time.

The viewport at design time has been moved and resized so that the items on the canvas are only partially enclosed in the viewport. At run time, the slide shows that the document window is initially the same size and shows the same area as the design-time viewport, although the user can usually resize this window. The canvas in this example is larger than the viewport. The canvas must completely contain all of its items, or you are unable to run the form.

There is an additional window, the MDI (Multiple Document Interface) window, that at run time displays the menu and toolbar, if any. It also displays the message and status areas, in addition to any document windows that may be open.

Using a Content Canvas

- A content canvas is the “base” canvas.
- Its view occupies the entire window.
- It is the default canvas type.
- Each window should have at least one content canvas.

ORACLE

11 - 5

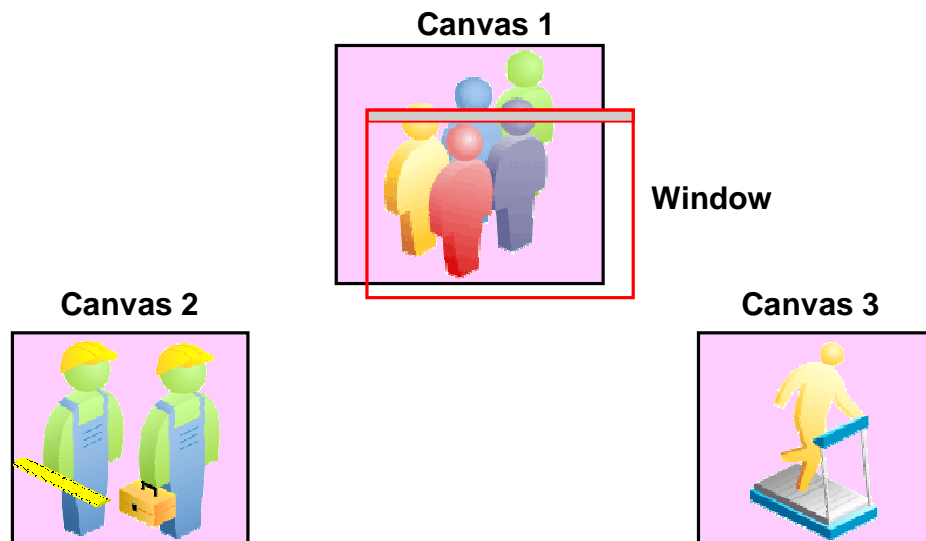
Copyright © 2009, Oracle. All rights reserved.

Using a Content Canvas

Forms Builder offers different types of canvases. A content canvas is the base canvas that occupies the entire content pane of the window in which it is displayed. The content canvas is the default canvas type. Most canvases are content canvases.

Note: Each item in a form must refer to no more than one canvas. An item displays on the canvas to which it is assigned, through its Canvas property. If the Canvas property for an item is left unspecified, that item is said to be a Null-canvas item and will not display at run time.

Relationship Between Windows and Content Canvases



11 - 6

Copyright © 2009, Oracle. All rights reserved.

ORACLE

Relationship Between Windows and Content Canvases

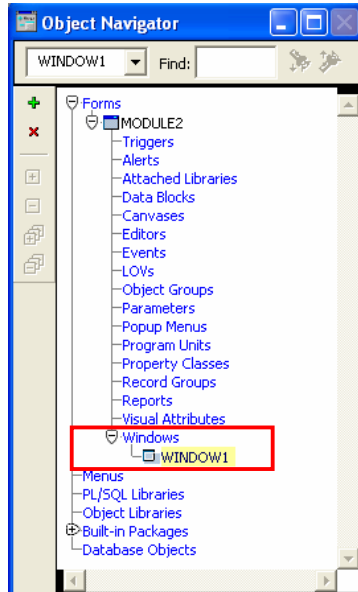
You must create at least one content canvas for each window in your application. When you run a form, only one content canvas can be displayed in a window at a time, even though more than one content canvas can be assigned to the same window at design time.

At run time, a content canvas always completely fills its window. As the user resizes the window, Forms resizes the canvas automatically. If the window is too small to show all items on the canvas, Forms automatically scrolls the canvas to bring the current item into view.

The slide shows three canvases and one window. All three canvases are assigned to the one window. The first canvas is displayed in the window. The other canvases are not currently displayed; however, if a user navigates to an item on one of these canvases, it appears in the window and the first canvas is no longer visible.

Note: You can assign multiple content canvases to a window; however, only one content canvas can be displayed in a window at a time.

Using the Default Window



WINDOW1 is created by default with each new form module.

- It is modeless.
- You can delete, rename, or change its attributes.

ORACLE

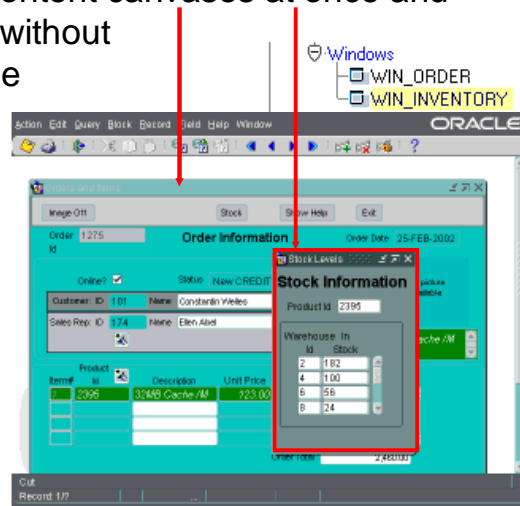
Using the Default Window

When you create a new form module, Forms Builder creates a new window implicitly. Thus, each new form module has one predefined window, which is called WINDOW1. You can delete or rename WINDOW1, or change its attributes.

The screenshot in the slide shows that the default window is the only object that is part of a new form.

Displaying a Form Module in Multiple Windows

- Use additional windows to:
 - Display two or more content canvases at once and switch between them without replacing the initial one
 - Modularize form contents
 - Take advantage of the window manager
- Two types of windows:
 - Modal
 - Modeless



Displaying a Form Module in Multiple Windows

Creating additional windows provides the ability to do the following:

- Display two or more content canvases at once
- Switch between canvases without replacing the initial one
- Modularize the form contents into multiple layouts
- Take advantage of window manager functionality, such as allowing the user to resize or close a window

Types of Windows

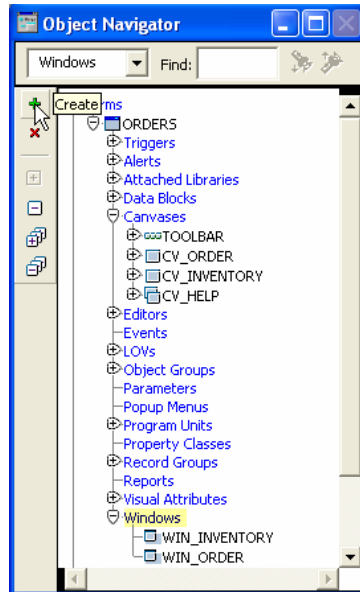
You can create two types of windows:

- A *modal window* is a restricted window that the user must respond to before moving the input focus to another window. Modal windows must be dismissed before control can be returned to a modeless window. They become active as soon as they are displayed, and require that you define a means of exit or dismissal.
- A *modeless window* is an unrestricted window that the user can exit freely. Modeless windows, which are the default window type, can be displayed at the same time as multiple other modeless windows. They are not necessarily active when displayed.

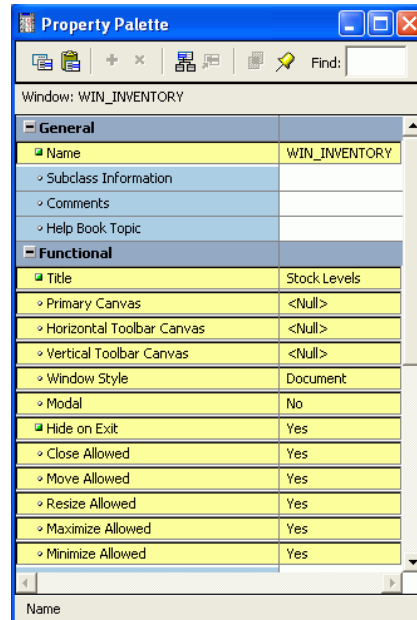
The screenshot shows a form's main window displaying orders, with an additional modeless window displaying inventory levels of a selected product.

Creating a New Window

Object Navigator: Click Create with the Windows node selected.



Property Palette: Set properties.



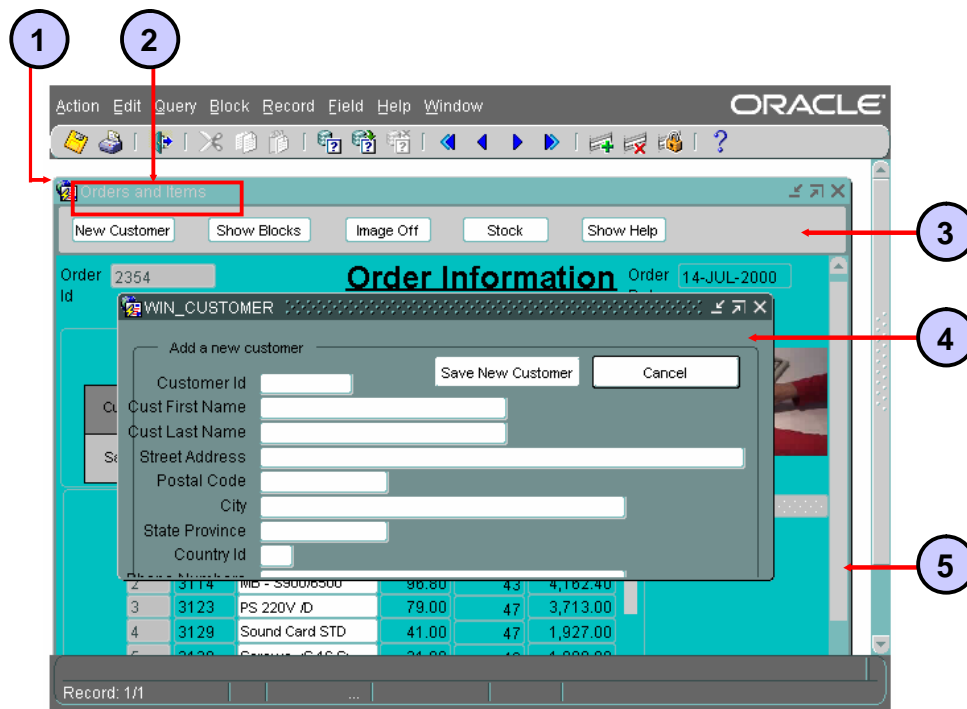
Creating a New Window

To create a new window, perform the following steps:

1. Select the Windows node in the Object Navigator.
2. Click Create. A new window entry displays, with a default name of WINDOWXX.
3. If the Property Palette is not already displayed, double-click the window icon to the left of the new window entry.
4. Set the window properties according to your requirements. The screenshot at the right of the slide shows the Property Palette of a window.

Note: For your new window to display, you must specify its name in the Window property of at least one canvas. To display a console to end users, set the form-level property Console Window to the window in which you want to display the console. To hide the console, set the property to <Null>.

Setting Window Properties



Setting Window Properties

You set properties for windows to determine their behavior and appearance. The following are some of the properties, which correspond to the numbered areas in the slide screenshot:

1. **X/Y Position:** Specifies the location of the window within the containing window
2. **Title:** The title to be displayed; if not specified, uses the name indicated by the window Name property
3. **Horizontal/Vertical Toolbar Canvas:** Specifies the toolbar canvas to be displayed horizontally across the top or vertically to the left of the window; selected from all horizontal/vertical toolbar canvases assigned to this window
4. **Modal:** Specifies if the window is modal, requiring the user to dismiss the window before any other user interaction can continue, such as the Add Customer window in the slide, where users must dismiss the window to continue in the Orders form
5. **Show Horizontal/Vertical Scroll Bar:** Specifies whether a horizontal or vertical scroll bar should display in the window
6. **Hide on Exit:** Indicates whether Forms hides the window automatically when the end user navigates to an item in another window (not shown in the slide)

For a description of other properties that affect the behavior of windows, click the property in the Property Palette and press F1.

Using GUI Hints

- GUI hints are recommendations to the window manager about appearance and functionality of a window.
- If the window manager supports a specific GUI hint and its property is set to Yes, it will be used.
- Functional properties for GUI hints:
 - Close Allowed
 - Move Allowed
 - Resize Allowed
 - Maximize Allowed
 - Minimize Allowed
 - Inherit Menu

ORACLE

11 - 11

Copyright © 2009, Oracle. All rights reserved.

Using GUI Hints

There are certain properties under the Functional group that enable you to make recommendations about window appearance and functionality:

- **Close Allowed:** Enables the mechanism for closing the window (Forms responds to a user's attempts to close the window by firing a When-Window-Closed trigger.)
- **Move/Resize Allowed:** Specifies whether a user can move and resize the window
- **Maximize/Minimize Allowed:** Specifies whether a user can zoom or iconify the window
- **Inherit Menu:** Specifies whether the window displays the current form menu

Displaying a Form Module on Multiple Layouts

PROPERTIES:

Canvas
CV_ORDER
Window:
WIN_ORDERS

Canvas
CV_INVENTORY
Window:
WIN_INVENTORY

The screenshot shows an Oracle Forms application with two overlapping windows. The background window, titled 'WIN_ORDERS', displays a 'CV_ORDER' canvas. It features a menu bar (Action, Edit, Query, Block, Record, Field, Help, Window) and a toolbar. Below the toolbar, there are input fields for 'Sales Rep: ID 155' and 'Name'. A table lists product items with columns 'Item#', 'Id', and 'Description'. The foreground window, titled 'WIN_INVENTORY', displays a 'CV_INVENTORY' canvas. It shows 'Stock Information' for a selected product (ID 3129). This window includes a 'Product Id' field and a table for 'Warehouse In' with columns 'Id' and 'Stock'.

Item#	Id	Description
1	3106	KB 101/EN
2	3114	MB - S900/650
3	3123	PS 220V /D
4	3129	Sound Card

Warehouse In	Id	Stock
2	198	
4	174	
6	150	
8	126	

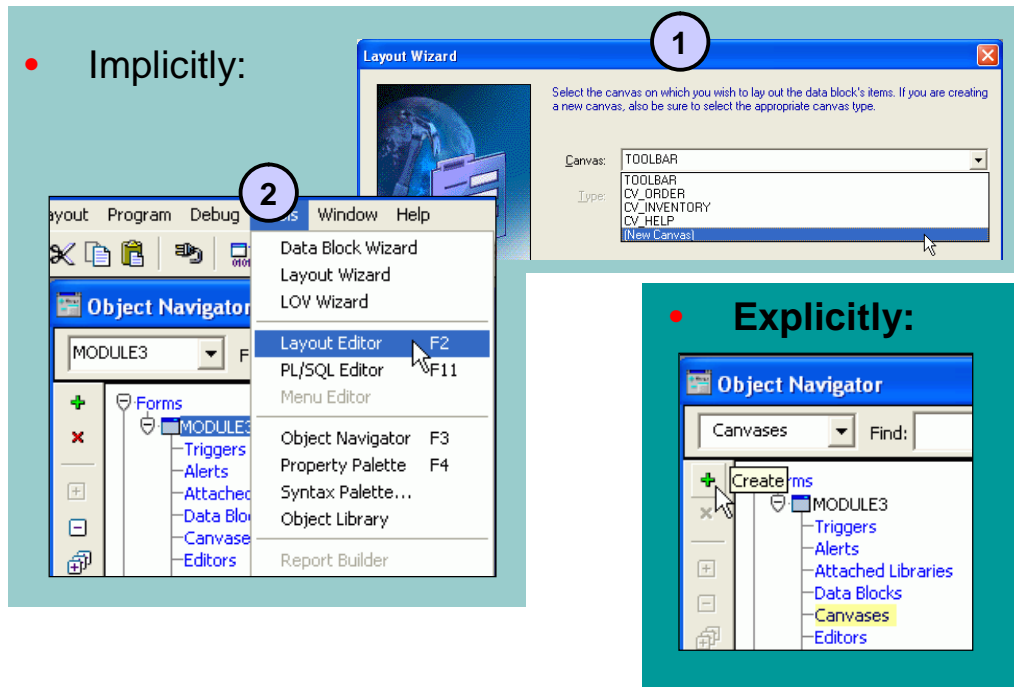
ORACLE

Displaying a Form Module on Multiple Layouts

You can have more than one content canvas in your Forms application. However, remember that only one content canvas can be displayed in a window at one time. To display more than one content canvas at the same time, you can assign each content canvas to a different window.

Now you can display the form module on multiple layouts or surfaces. The screenshot shows two content canvases that are displayed simultaneously, each in a separate window. One window (WIN_ORDERS) displays the canvas containing orders and order items (CV_ORDER). The other window (WIN_INVENTORY) displays the canvas showing inventory levels of a selected product (CV_INVENTORY.)

Creating a New Content Canvas



11 - 13

Copyright © 2009, Oracle. All rights reserved.

ORACLE

Creating a New Content Canvas

The creation of a content canvas can be implicit or explicit.

Implicitly Creating a Content Canvas

There are two ways of implicitly creating a new content canvas:

1. **Layout Wizard:** When you use the Layout Wizard to arrange data block items on a canvas, the wizard enables you to select a new canvas on its Canvas page. In this case, the wizard creates a new canvas with a default name of CANVASXX. (See the top-right screen shot in the slide.)
2. **Layout Editor:** When there are no canvases in a form module and you invoke the Layout Editor, as shown in the screenshot at the left of the slide, Forms Builder creates a default canvas on which you can place items.

Explicitly Creating a Content Canvas

You can create a new content canvas explicitly by using the Create icon in the Object Navigator, as shown in the screenshot at the bottom-right of the slide.

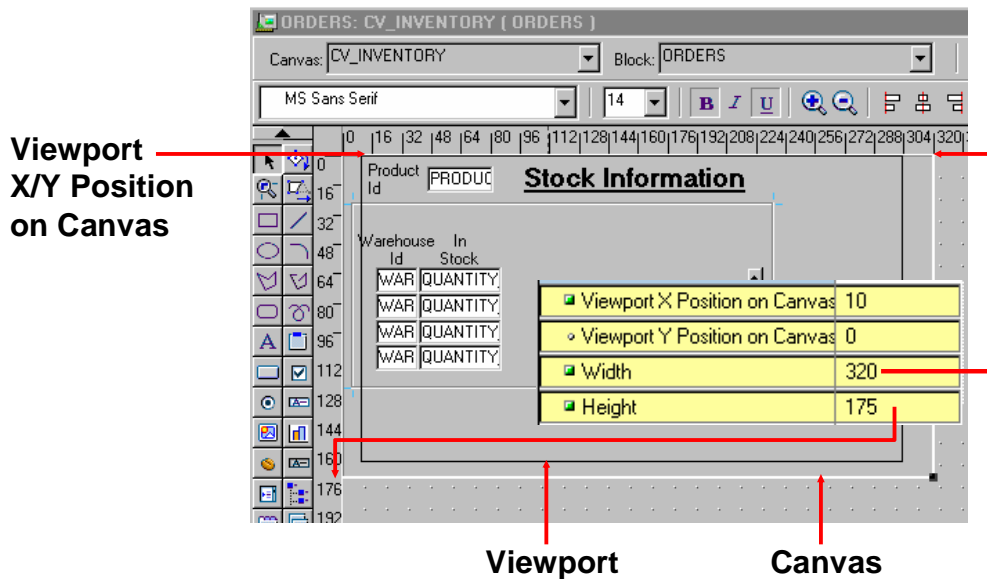
Creating a New Content Canvas (continued)

To explicitly create a new content canvas, perform the following steps:

1. Click the Canvases node in the Object Navigator.
2. Click the Create icon. A new canvas entry appears with a default name of CANVASxx.
3. If the Property Palette is not already displayed, select the new canvas entry. Then select Tools > Property Palette.
4. Set the canvas properties according to your requirements.

Note: Double-clicking the icon for a canvas in the Object Navigator invokes the Layout Editor instead of the Property Palette.

Setting Content Canvas Properties



ORACLE

11 - 15

Copyright © 2009, Oracle. All rights reserved.

Setting Content Canvas Properties

You can set properties to determine how the canvas is to be displayed. Several properties in the Physical group are shown in the slide: Width, Height, and Viewport X/Y Position on Canvas. For a canvas to display at run time, its Window property must also be specified and its Visible property must be set to Yes.

In the General group, you can choose the Canvas Type. (This lesson explains the content canvas. For information about other canvas types, refer to the lesson titled “Working with Other Canvas Types.”)

In the Functional group, you can set “Raise on Entry” to affect whether the canvas is always brought to the front of the window when the user navigates to an item on the canvas. You use this property when the canvas is displayed in the same window with other canvases. Forms always ensures that the current item is visible. Even if “Raise on Entry” is set to No, Forms will bring the canvas to the front of the window if the user navigates to an item on the canvas that is hidden behind other canvases.

Performance Tip: To reduce the time that is needed to display the initial screen, keep the number of items initially displayed to a minimum. You can hide elements, such as canvases, that are not immediately required. To do this, set the canvas “Raise on Entry” property to Yes, and set Visible to No.

Summary

In this lesson, you should have learned that:

- Windows can display multiple content canvases, but can display only one canvas at a time
- Content canvases are displayed only in the window to which they are assigned
- You must assign at least one content canvas to each window in your application
- You create windows in the Object Navigator; one is created by default with each new module
- You create canvases in the Object Navigator, by using the Layout Wizard, or by invoking the Layout Editor in a module without a canvas
- You can display multiple layouts by assigning canvases to different windows

ORACLE

11 - 16

Copyright © 2009, Oracle. All rights reserved.

Summary

In this lesson, you should have learned:

- About the relationship between windows and content canvases
- How to create a new window
- How to create a new content canvas
- How to display a form module on multiple layouts by displaying each canvas in a separate window

Practice 11: Overview

This practice covers the following topics:

- Changing a window size, position, name, and title
- Creating a new window
- Displaying data block contents in the new window

ORACLE

11 - 17

Copyright © 2009, Oracle. All rights reserved.

Practice 11: Overview

In this practice, you customize windows in your form modules. You resize the windows to make them more suitable for presenting canvas contents. You also create a new window to display the contents of the INVENTORIES block.

- Change the size and position of the window in the Customers form. Change its name and title. Save and run the form.
- Modify the name and title of the window in the Orders form.
- Create a new window in the Orders form. Make sure the contents of the INVENTORIES block display in this window. Save and run the form.

12

Working with Other Canvas Types

ORACLE[®]

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe the different types of canvases and their relationships to each other
- Identify the appropriate canvas type for different scenarios
- Create an overlay effect by using stacked canvases
- Create a toolbar
- Create a tabbed interface

ORACLE

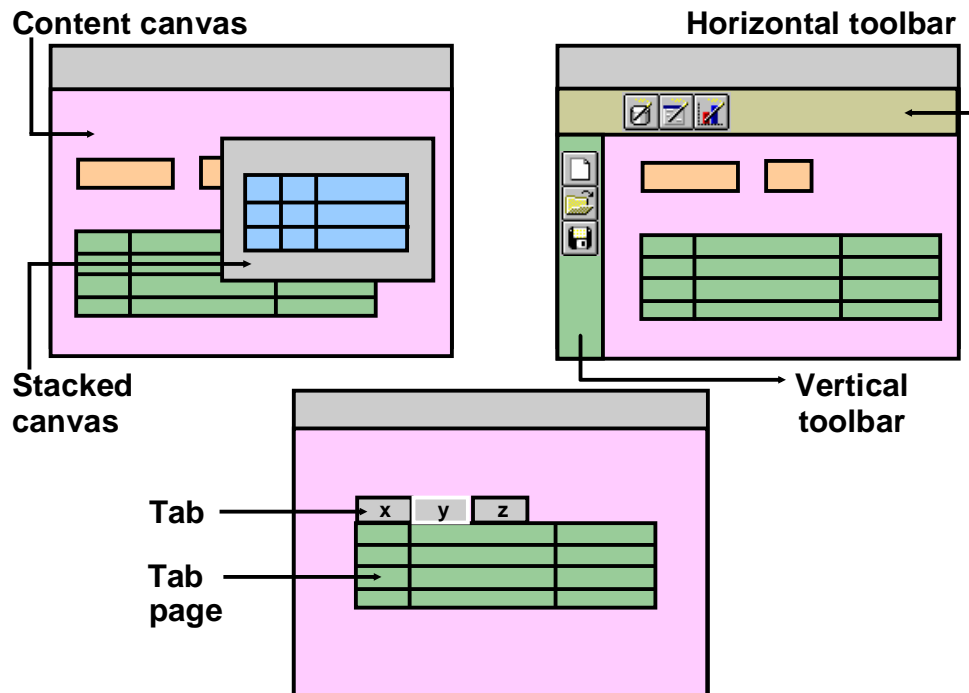
12 - 2

Copyright © 2009, Oracle. All rights reserved.

Lesson Aim

In addition to the content canvas, Oracle Forms Builder enables you to create three other canvas types. This lesson introduces you to the stacked canvas, which is ideal for creating overlays in your application. It also presents the toolbar canvas and the tabbed canvas, both of which enable you to provide a user-friendly GUI application.

Overview of Canvas Types



Overview of Canvas Types

In addition to the content canvas, Forms Builder provides three other types of canvases:

- **Stacked canvas:** The slide shows a smaller stacked canvas appearing on top of the content canvas, obscuring part of the content canvas.
- **Toolbar canvas:** The slide shows a vertical toolbar to the left of the canvas and a horizontal toolbar at the top of the canvas; toolbars do not obscure any of the content canvas.
- **Tab canvas:** The slide shows that the user can select different tabs to display different information on the canvas.

When you create a canvas, you specify its type by setting the Canvas Type property. The type determines how the canvas is displayed in the window to which it is assigned.

What Is a Stacked Canvas?

- The stacked canvas is displayed on top of a content canvas.
- It shares a window with a content canvas.
- Its size is:
 - Usually smaller than the content canvas in the same window
 - Determined by viewport size
- It is created in:
 - Layout Editor (usually easier)
 - Object Navigator

ORACLE

12 - 4

Copyright © 2009, Oracle. All rights reserved.

What Is a Stacked Canvas?

A *stacked canvas* is displayed on top of, or stacked on, the content canvas that is assigned to a window. It shares a window with a content canvas and any number of other stacked canvases. Stacked canvases are usually smaller than the window in which they display.

You can create a stacked canvas in either of the following:

- Layout Editor (usually easier because you can set properties visually)
- Object Navigator

Determining the Size of a Stacked Canvas

Stacked canvases are typically smaller than the content canvas in the same window. You can determine the stacked canvas dimensions by setting the Width and Height properties. You can determine the view dimensions of the stacked canvas by setting the Viewport Width and Viewport Height properties.

Typical Usage of a Stacked Canvas

If a data block contains more items than the window can display, Forms scrolls the window to display items that were initially not visible. This can cause important items, such as primary key values, to scroll out of view. By placing important items on a content canvas, and then placing the items that can be scrolled out of sight on a stacked canvas, the stacked canvas becomes the scrolling region, rather than the window itself.

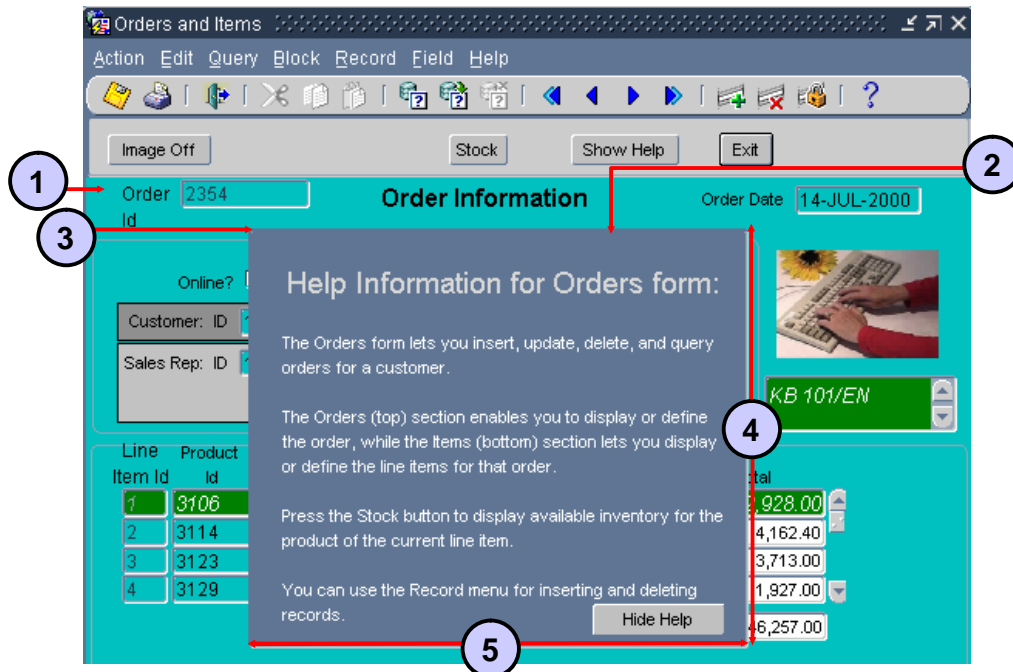
What Is a Stacked Canvas? (continued)

Typical Usage of a Stacked Canvas

With stacked canvases, you can achieve the following:

- Scrolling views
- Creating an overlay effect within a single window
- Displaying headers with constant information, such as company name
- Creating a cascading or a revealing effect within a single window
- Displaying additional information
- Displaying information conditionally
- Displaying context-sensitive help
- Hiding information

Stacked Canvas Elements

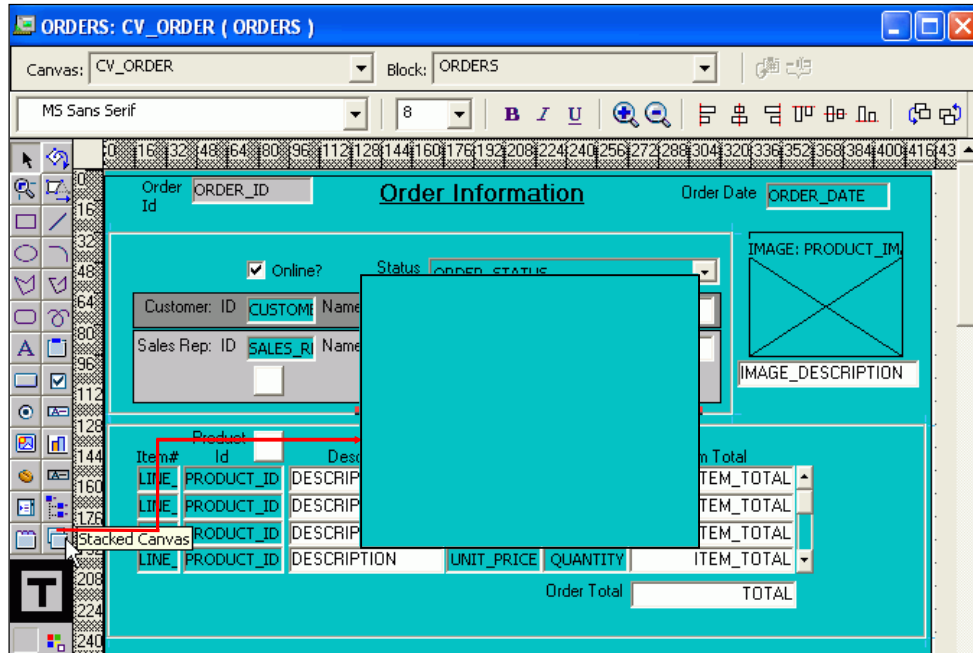


Stacked Canvas Elements

The stacked canvas is displayed on the content canvas. You can see the following elements in the screenshot in the slide:

1. Content canvas
2. Stacked canvas
3. Viewport X/Y position of the stacked canvas
4. Viewport height of the stacked canvas
5. Viewport width of the stacked canvas

Creating a Stacked Canvas



Creating a Stacked Canvas

How to Create a Stacked Canvas in the Layout Editor

It is usually easier to use the Layout Editor to create a stacked canvas because you can visually set its properties. To create a stacked canvas in the Layout Editor, perform the following steps:

1. In the Object Navigator, double-click the object icon for the content canvas on which you want to create a stacked canvas.
The Layout Editor is displayed.
2. Click the Stacked Canvas tool in the toolbar.
3. Click and drag in the canvas where you want to position the stacked canvas. The slide graphic pictures the Layout Editor immediately after this step has been done. At this point, the stacked canvas appears as a blank window on top of and obscuring part of the content canvas.
4. Open the Property Palette of the stacked canvas. Set the canvas properties according to your requirements (described later in this lesson).

Creating a Stacked Canvas (continued)

How to Create a Stacked Canvas in the Object Navigator

To create a stacked canvas in the Object Navigator, perform the following steps:

1. Select the Canvases node in the Object Navigator.
2. Click the Create icon.
A new canvas entry displays with a default name of CANVASXX.
3. If the Property Palette is not already displayed, select the new canvas entry and then select Tools > Property Palette.
4. Set the Canvas Type property to Stacked. Additionally, set the properties that are described later in this lesson according to your requirements.
5. Ensure that the stacked canvas is below the content canvas in the Object Navigator.

Note: To convert an existing content canvas to a stacked canvas, change its Canvas Type property value from Content to Stacked.

For the stacked canvas to display properly at run time, ensure that its position in the stacking order places it in front of the content canvas assigned to the same window. The stacking order of canvases is defined by the sequence in which they appear in the Object Navigator.

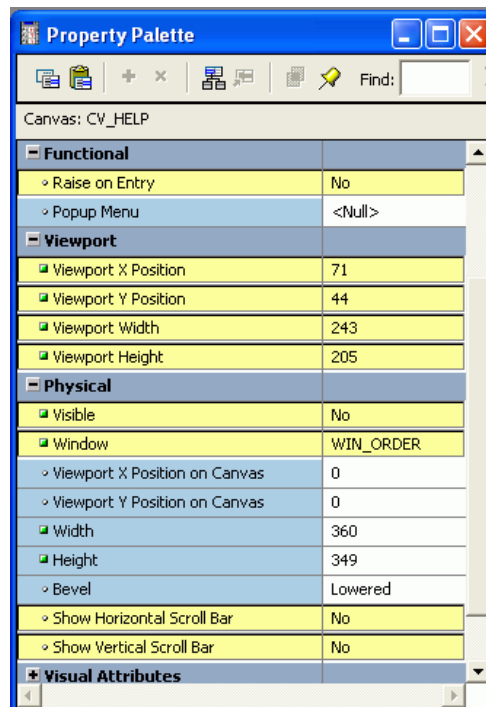
Displaying Stacked Canvases in the Layout Editor

You can display a stacked canvas as it sits over the content canvas in the Layout Editor. You can check the display position of stacked canvases by doing the following:

- Select View > Stacked Views in the Layout Editor. The Stacked/Tab Canvases dialog box is displayed, with a list of all the stacked canvases assigned to the same window as the current content canvas.
- Select the stacked canvases you want to display in the Layout Editor.

Note: Press Ctrl and click to clear a stacked canvas that was previously selected.

Setting Stacked Canvas Properties



ORACLE

12 - 9

Copyright © 2009, Oracle. All rights reserved.

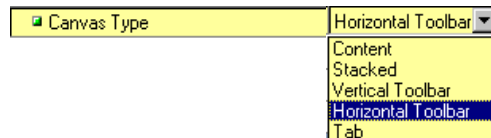
Setting Stacked Canvas Properties

There are several properties that you can set to affect the appearance and behavior of stacked canvases. The screenshot shows the Property Palette for a stacked canvas:

- **“Raise on Entry”:** Controls how Forms displays the canvas when the user or the application navigates to an item on the canvas
- **Viewport X/Y Position:** Specifies the point at which the upper-left corner of the stacked canvas is located in relation to the content canvas
- **Viewport Width/Height:** Determines the size of the stacked canvas displayed at run time
- **Visible:** Indicates whether the stacked canvas is initially displayed or visible; set to Yes or No to show or hide the stacked canvas
- **Window:** Specifies the window in which the canvas will be displayed
- **Show Horizontal/Vertical Scroll Bar:** Specifies whether scrollbars display with the canvas

What Is a Toolbar Canvas?

- Special type of canvas for tool items
- Two types:
 - Vertical toolbar
 - Horizontal toolbar
- Provide:
 - Standard look and feel
 - Alternative to menu or function key operation



ORACLE

12 - 10

Copyright © 2009, Oracle. All rights reserved.

What Is a Toolbar Canvas?

A *toolbar canvas* is a special type of canvas that you can create to hold buttons and other frequently used GUI elements.

Toolbar Types

- **Vertical toolbar:** Use a vertical toolbar to position all your tool items to the left of your window.
- **Horizontal toolbar:** Use a horizontal toolbar to position all your tool items and controls across the top of your window under the window's menu bar.

The screenshot in the slide shows that each of these toolbar types is available in the drop-down list for canvas types in the Property Palette for a canvas.

Uses and Benefits of Toolbars

Toolbar canvases offer the following advantages:

- Provide a standard look and feel across canvases displayed in the same window.
- Decrease form module maintenance time.
- Increase application usability.
- Create applications similar to others used in the same environment.
- Provide an alternative to menu or function key–driven applications.

Creating a Toolbar Canvas

1. Create new or modify the existing canvas.
 - a. Click Create in the Object Navigator.
 - b. Change Canvas Type.
 - c. Set other properties as needed.
2. Add functionality.
3. Resize the canvas (not the view).
4. Assign to window and/or form.

ORACLE

12 - 11

Copyright © 2009, Oracle. All rights reserved.

Creating a Toolbar Canvas

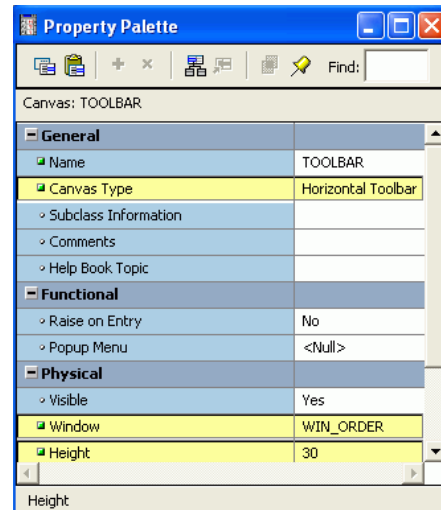
To create a Toolbar canvas, perform the following steps:

1. Create new or modify the existing canvas:
 - With the Canvases node selected, click the Create icon in the Object Navigator.
 - If the Property Palette is not already displayed, click the new canvas entry and select Tools > Property Palette.
 - Set the Canvas Type to either Horizontal or Vertical Toolbar.
2. Add functionality, usually with push buttons. To avoid problems with unwanted navigation, be sure to set the Mouse Navigate property of any toolbar buttons to No.
3. Resize: Use the Layout Editor to resize the toolbar canvas (not just the view) to the smallest size necessary to display its items.
4. Assign to window or form: In the Property Palette of the window or the form module, set the Horizontal or Vertical Toolbar Canvas properties.

When toolbars are placed on the form or window, the window manager controls shifting the content canvas downward or to the right in order to display the toolbar. You do not need to leave enough space in the content canvas for the placement of the toolbar.

Setting Toolbar Properties

- Canvas properties:
 - Canvas Type
 - Window
 - Width or Height
- Window properties:
 - Horizontal Toolbar Canvas
 - Vertical Toolbar Canvas
- Form module properties:
 - Form Horizontal Toolbar Canvas
 - Form Vertical Toolbar Canvas



Setting Toolbar Properties

After you create a toolbar canvas, you can set its properties as well as the properties of the associated window or form module:

- **Canvas Properties**
 - **Canvas Type:** Can be either Horizontal or Vertical Toolbar
 - **Window:** Specifies in which window the toolbar displays
 - **Width/Height:** Specifies the size of the canvas. The width of a horizontal toolbar is set to the width of the window (for example, content canvas). Likewise, the height of a vertical toolbar is set to the height of the window.
- **Window Property**
 - Horizontal/Vertical Toolbar Canvas: Identifies the horizontal/vertical toolbar to be displayed in this window
- **Form Module Property**
 - Form Horizontal/Vertical Toolbar Canvas: Identifies the horizontal/vertical toolbar to be displayed in the Multiple Document Interface (MDI) window (when the useSDI run-time parameter is set to No)

Using an MDI Toolbar

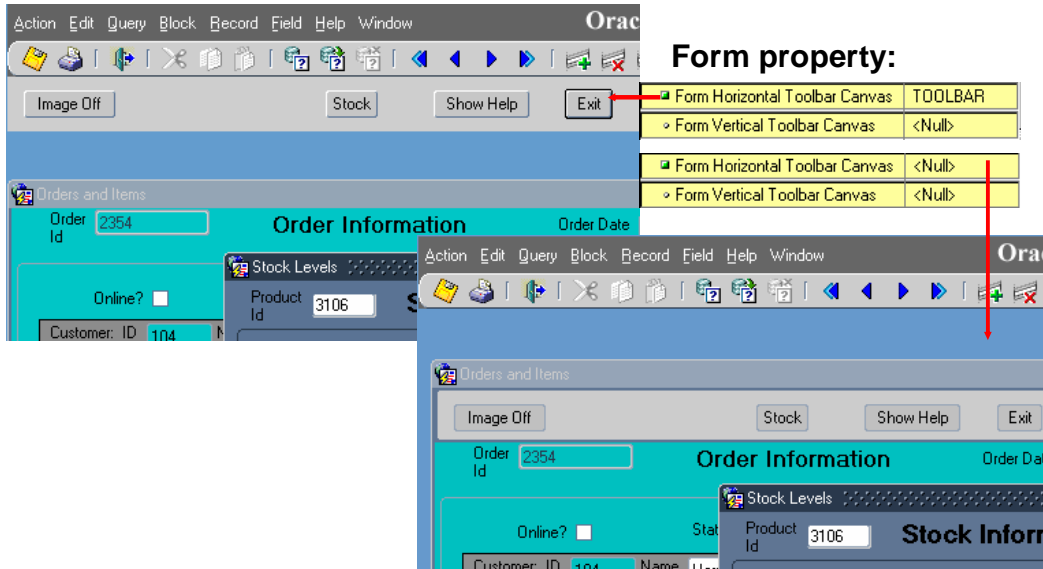
Run-time parameter:
otherparams=useSDI=no

Window property:

Horizontal Toolbar Canvas	TOOLBAR
Vertical Toolbar Canvas	<Null>

Form property:

Form Horizontal Toolbar Canvas	TOOLBAR
Form Vertical Toolbar Canvas	<Null>
Form Horizontal Toolbar Canvas	<Null>
Form Vertical Toolbar Canvas	<Null>



Using an MDI Toolbar

You can attach the toolbar to individual windows, or to the form itself. Attaching a toolbar to a form provides an MDI toolbar, so that you do not need to create more than one toolbar for a Forms application that uses multiple windows. If you display a toolbar in the MDI window, the same toolbar will not be duplicated in the individual windows of the form.

Whether an MDI toolbar is displayed is determined by a combination of Form properties and the useSDI run-time parameter, which you set as part of otherparams.

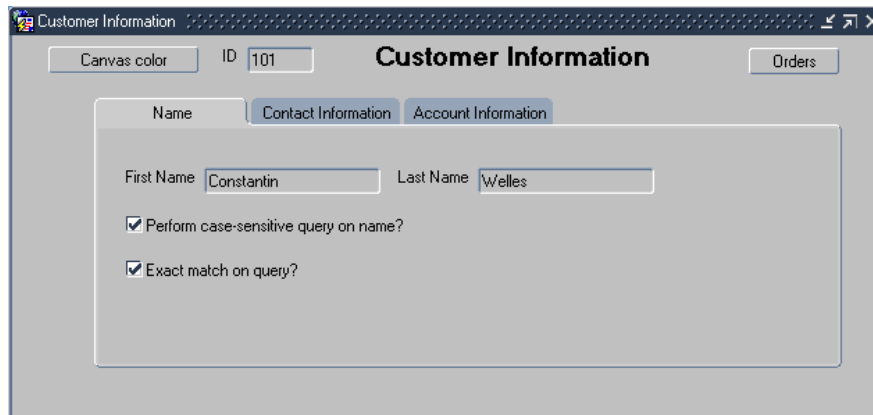
Setting	Setting Type	Effect
Form Horizontal or Vertical Toolbar Canvas	Form Property	If useSDI=no, displays MDI toolbar at the top or left of the MDI window
otherparams=useSDI=no	Run-time setting	Displays the MDI window and uses the MDI toolbar if set as a Form property
otherparams=useSDI=yes	Run-time setting	MDI window not displayed; Form Horizontal or Vertical Toolbar setting, if any, has no effect

Using an MDI Toolbar (continued)

The screenshots illustrate how toolbar canvases appear when `useSDI` is set to `no`:

- The screenshot at the left shows using an MDI toolbar by setting the Form Horizontal Toolbar Canvas to the name of the toolbar canvas. The toolbar canvas appears at the top of the form MDI window, but not on individual windows.
- The screenshot at the bottom shows that when an MDI toolbar is not defined, the toolbar appears on each individual window, rather than on the MDI window.

What Is a Tab Canvas?



- Enables you to organize and display related information on separate tabs
- Consists of one or more tab pages
- Provides easy access to data

ORACLE

12 - 15

Copyright © 2009, Oracle. All rights reserved.

What Is a Tab Canvas?

A *tab canvas* is a special type of canvas that enables you to organize and display related information on separate tabs. Like stacked canvases, tab canvases are displayed on top of a content canvas. The screenshot in the slide shows a content canvas for Customer Information that has a tabbed canvas displayed over it. Each tab page is labeled with the type of information it contains (name, contact, and account information), but the customer ID is on the content canvas.

What Is a Tab Page?

A *tab page* is a subobject of a tab canvas. Each tab canvas is made up of one or more tab pages. A tab page displays a subset of the information in the entire tab canvas. Each tab page has a labeled tab that end users can click to access information on the page.

Each tab page occupies an equal amount of space on the tab canvas.

Uses and Benefits of Tab Canvases

You can use tab canvases to:

- Create an overlay effect within a single window
- Display large amounts of information on a single canvas
- Hide information and easily access it by clicking the tab

Creating a Tab Canvas

1. Create in:
 - Object Navigator
 - Layout Editor (usually easier)
2. Define tab pages.
3. Place items on tab pages.

ORACLE

12 - 16

Copyright © 2009, Oracle. All rights reserved.

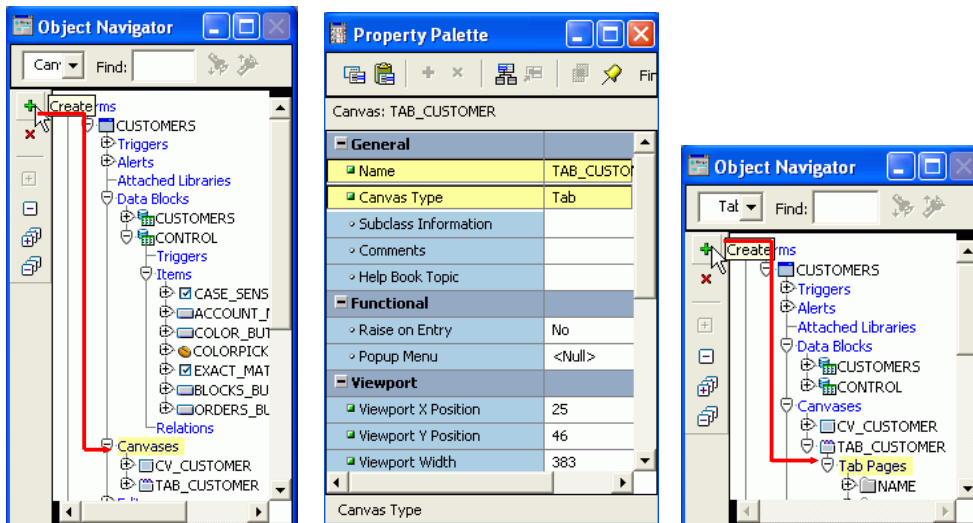
Creating a Tab Canvas

To create a functional tab canvas, you must:

1. Create an empty tab canvas in either of the following:
 - Object Navigator
 - Layout Editor; it may be easier to use the Layout Editor to create a tab canvas because you can visually set its properties
2. Define one or more tab pages for the tab canvas.
3. Place items on the tab pages.

Note: Although it is possible to get rid of a content canvas after creating a tab canvas, it is still highly recommended that there be at least one content canvas for each window in your Forms application.

Creating a Tab Canvas in the Object Navigator



Create new canvas.

Set Canvas Type.

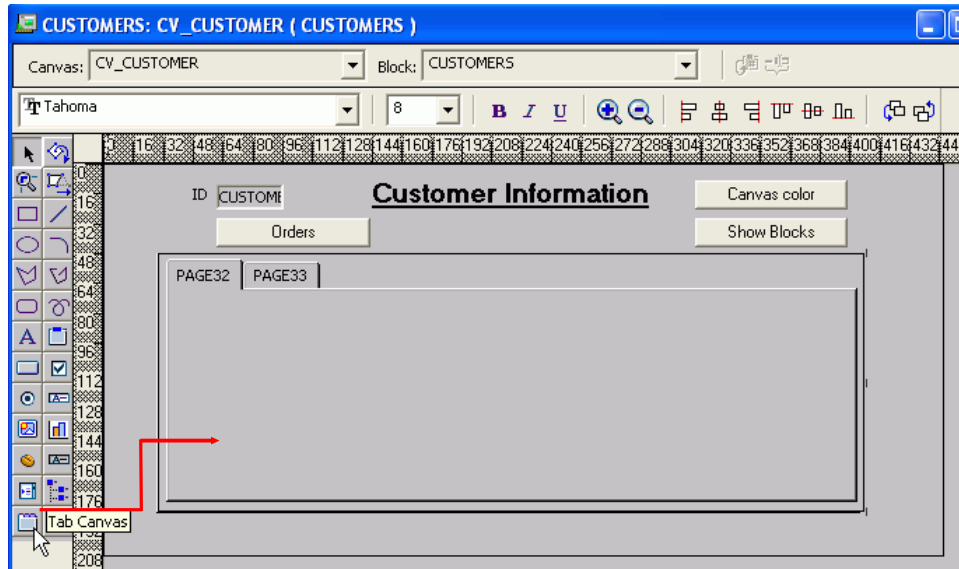
Create Tab Pages.

Creating a Tab Canvas in the Object Navigator

To create a tab canvas in the Object Navigator, perform the following steps:

1. Select the Canvases node in the Object Navigator.
2. Click the Create icon. A new canvas entry is displayed.
3. If the Property Palette is not already displayed, select the new canvas entry and select Tools > Property Palette.
4. Set the Canvas Type property to Tab. Additionally, set the canvas properties according to your requirements (described later in the lesson).
5. Expand the canvas node in the Object Navigator. The Tab Pages node is displayed.
6. Click the Create icon, as shown in the screenshot at the right of the slide. A tab page displays in the Object Navigator, with a default name of PAGEXX. The Property Palette takes on its context.
7. Set the tab page properties according to your requirements (described later in the lesson).
8. Create additional tab pages by repeating steps 6 and 7.

Creating a Tab Canvas in the Layout Editor



ORACLE

12 - 18

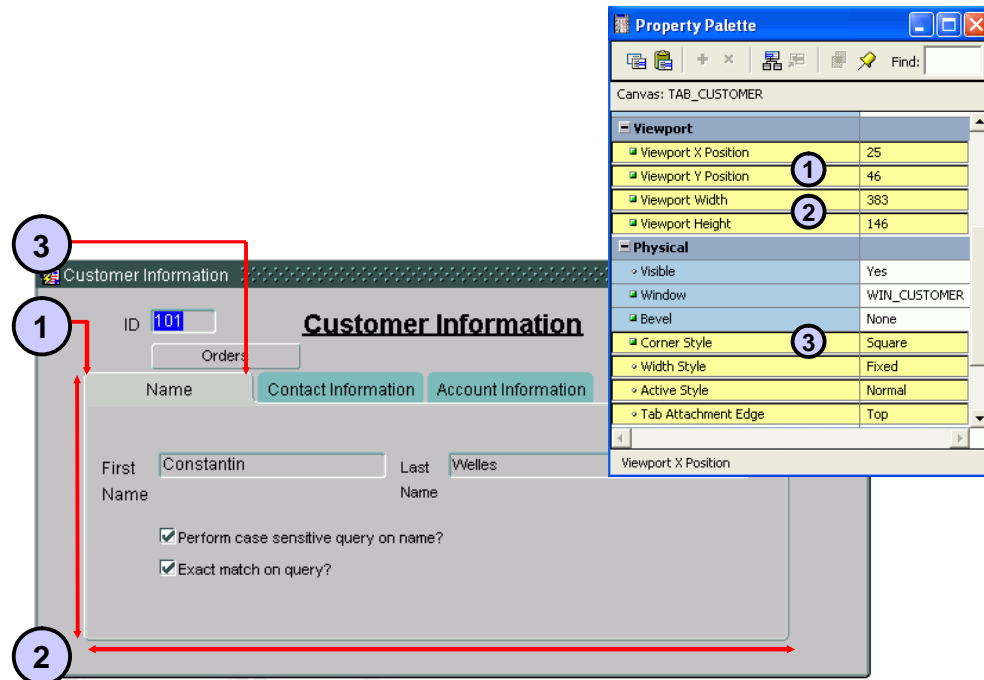
Copyright © 2009, Oracle. All rights reserved.

Creating a Tab Canvas in the Layout Editor

To create a tab canvas in the Layout Editor, perform the following steps:

1. In the Object Navigator, double-click the object icon for the content canvas on which you want to create a tab canvas.
The Layout Editor is displayed.
2. Click the Tab Canvas tool in the toolbar.
3. Click and drag in the canvas where you want to position the tab canvas.
Forms Builder creates a tab canvas with two tab pages by default. The screenshot in the slide shows the Layout Editor immediately after this step has been performed; it shows the tab canvas on top of the content canvas, with two default tab pages.
4. Open the Property Palette of the tab canvas. Set the canvas properties according to your requirements, as described on the next page of this lesson.
5. Create additional tab pages, if needed, in the Object Navigator.
6. Set the tab page properties according to your requirements, as described on the next page of this lesson.

Setting Tab Canvas, Tab Page, and Item Properties



Setting Tab Canvas, Tab Page, and Item Properties

After you create a tab canvas and its tab pages, you must set some properties for both of these objects. Place items on a tab page by setting the necessary item properties.

- **Tab Canvas Properties** (the numbered properties are illustrated in both the Property Palette and the run-time form):
 1. **Viewport X/Y Position:** Specifies the point at which the upper-left corner of the tab page is located in relation to the content canvas
 2. **Viewport Width/Height:** Defines the size of the tab canvas
 3. **Corner Style:** Specifies the shape of labeled tabs (Chamfered, Square, or Rounded); the screenshot in the slide shows a Square corner style.
 - **Tab Attachment Edge:** Specifies where tabs are attached (Top, Bottom, Left, Right, Start, or End)
 - **Width Style:** Specifies whether the width of the tab will vary with the length of the label
 - **Active Style:** Specifies whether the label displays as bold when the tab page is active
- **Tab Page Property:** Specifies the label to be displayed on the tab page's tab
- **Item Properties:**
 - **Canvas:** The tab canvas on which the item will be displayed
 - **Tab Page:** The tab page on which the item will be displayed

Placing Items on a Tab Canvas

- Place items on each tab page for user interaction.
- Set the item properties:
 - Canvas
 - Tab Page

ORACLE

12 - 20

Copyright © 2009, Oracle. All rights reserved.

Placing Items on a Tab Canvas

After you create a tab canvas and related tab pages, you place items on the tab pages that the end users can interact with at run time:

- Open the Property Palette of the item.
- Set the item's Canvas and Tab Page properties to the desired tab canvas and tab page.

Note: Display the tab canvas as it sits on top of the content canvas, by selecting View > Stacked View in the Layout Editor.

Performance Tip: The time taken to load and initialize a tab canvas at run time is related to all objects on the canvas and not just to those initially visible. To improve this loading and initialization time, you can use a tab canvas to get the tabbed effect and place stacked canvases on the tab pages, adding the items and other components on the stacked canvases. By doing this, the components are not rendered until they are needed.

Summary

In this lesson, you should have learned that:

- You can use canvas types other than content:
 - Stacked
 - Toolbar
 - Tab
- You can create these in the Object Navigator and change the canvas type, and then set properties
- You can create stacked or tab canvases with the appropriate tool in the Layout Editor
- You can attach a toolbar canvas to a single window, or to the entire form if using MDI
- After creating a tab canvas, you create tab pages and place related items on them

ORACLE

12 - 21

Copyright © 2009, Oracle. All rights reserved.

Summary

In this lesson, you should have learned to:

- Use a stacked canvas to:
 - Create an overlay effect
 - Create a cascading or revealing effect within a single window
 - Display addition information
 - Conditionally display or hide information
 - Provide context-sensitive help
- Create a toolbar:
 - Display on top or to the left of a content canvas
 - Provide buttons or other frequently used GUI elements
 - Enforce a standard look and feel across canvases displayed in the same form or window
- Create a tabbed canvas to organize and display related information on multiple tab pages

Practice 12: Overview

This practice covers the following topics:

- Creating a toolbar canvas
- Creating a stacked canvas
- Creating a tab canvas
- Adding tab pages to the tab canvas

ORACLE

12 - 22

Copyright © 2009, Oracle. All rights reserved.

Practice 12: Overview

In this practice session, you will create different types of canvases: stacked canvas, toolbar canvas, and tab canvas.

- Create a horizontal toolbar canvas in the Orders form. Create new buttons in the CONTROL block, and place them on the horizontal toolbar. Save and run the form.
- Create a stacked canvas in the Orders form to add some help text. Position the canvas in the center of the window. Create a button in the CONTROL block. This button will be used later to display the stacked canvas. Add help text on the stacked canvas. Save and run the form.
- Create a tab canvas in the Customers form. Create three tab pages on this canvas, and make sure that each tab page displays the appropriate information. Save and run the form.